Cross-Layer Optimization for Wireless Communication Networks

A dissertation submitted by

Yuping Dong

In partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

in

Electrical Engineering

TUFTS UNIVERSITY
February 2016

Advisor:
Prof. Chorng Hwa Chang

Abstract

Cross-Layer Design (CLD) has been widely studied and developed to optimize resource allocation decisions in wireless communication networks. Various design schemes have been proposed for the purpose of enhancing performance of resource restricted, error-prone wireless networks. This dissertation presents two CLDs for wireless networks; one promotes network energy efficiency, the other provides less resource-intensive optimization and maximizes system reward.

The energy efficient routing algorithm keeps records of wireless sensors' energy levels, utilizes this information to determine routing decisions at an upper layer. With this algorithm, network power consumption is distributed among all sensors, which in turn prolongs the lifetime of the network. Simulations and comparisons shown in the paper of energy conserving routing algorithm demonstrate that this algorithm gains more than 40% improvement on energy saving over the energy-efficient m-coverage and n-connectivity routing algorithm. This algorithm is a CLD in which the network layer gets information about the physical layer, including remaining energy and transmission power, to make better routing decisions in real time.

By contrast, the autonomous CLD accumulates information from lower layers, passes it to upper layers. And vice versa. Each layer only communicates with its neighbors. Each layer has its own optimizer which can be run in real-time to determine the best transmission parameters in order to maximize the wireless user's system utility.

This optimization accounts for different Quality of Service (QoS) requirements involving different communication protocols. Comparing to centralized optimization, which uses a centralized optimizer to gather the environment dynamics of all layers before calculating best transmission strategies for each layer, this solution requires less computation time on the resource constrained wireless devices and adapts to various data sources quickly. This theoretical model describes many practical networks. Ns-3 simulations of this algorithm for a Wi-Fi network demonstrate improved network performance, including maximizing network throughput, which is almost doubled over an unmanaged solution, lowering transmission cost by more than one half and reducing transmission delay. Comparing the improved autonomous CLD with the original design proposed by F. Fu et al., the improved version has more robust performance with more transmission cost tolerance and faster optimization calculation.

Acknowledgments

My deep gratitude goes first to Prof. Chorng Hwa Chang, who expertly guided me through my Ph.D education and who shared the excitement of every single discovery. His continuous encouragement, guidance and support were invaluable and indispensable to the completion of this dissertation. His concern and advices helped make my oversea life unstressed and even enjoyable.

Special thanks go to Prof. Alva L. Couch, who generously offered to co-advise my dissertation during the last year of completion. He is always there when I have questions/problems and gives great suggestions. His superior academic skills and great passion for networking helped elevate my dissertation to a higher level.

I would like to thank Prof. Joeseph P. Noonan and Prof. Byung Guk Kim for their time and dedication. They offered insightful suggestions on this work.

My appreciation to all the existing and former group members of Tufts Wireless Laboratory. Numerous great ideas and inspirations came from group discussions.

Many thanks to Miriam L. Santi, who never stopped encouraging me. I always get motivation after talking with her.

I am deeply grateful for having a happy family full of love. My sweetest daughter and my beloved husband keep me going even when I am facing challenges. Our family is becoming even happier with a new member coming soon.

Last but not least, I would like to thank my sister, Yurong Dong and my parents, my mom in China, Mama Anne and Papa George, for their endless love and support. Whenever I feel sad, I know they are always there for me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

With advanced technology, today's wireless networks handle a mixture of real-time traffic such as high definition voice/video streaming, multimedia teleconferencing, online games, and data traffic such as Internet web browsing, text messaging and file transfers. All of these applications have very high performance requirements on wireless networks with diverse QoS guarantees [36]. The traditional layered abstraction, the Open Systems Interconnection (OSI) model, partitions a communication system into seven layers, with each lower layer provides services to the layer above it. Each abstraction layer runs an individual protocol and makes independent decisions on selecting protocol parameters. When designing lower layer protocols, the designers do not consider the characteristics of specific multimedia applications, whereas the designers of higher layers do not take into account of the status of lower layers. This design manner prevents the protocol under optimization from maximizing the system performance in real time by not adapting to the parameters of the multimedia applications and the characteristics of the time-varying wireless devices and

channels.

Modern Modulation and Coding Schemes (MCS) eliminate interference among multi-carrier data transmissions. However, the fact that mobile stations move around constantly, and the multipath reflections of transmitted signals make the communication medium unstable. The power of standalone wireless devices depletes quickly without energy saving strategies. Cross-layer optimization jointly analyzes, selects and adapts different strategies available at various OSI layers in order to deliver a required data quality at the receiver by expediting the wireless network performance by increasing network throughput, lowering transmission latency, and reducing device power consumption.[41]

1.2 Cross-Layer Optimization Methods

As wireless communications and networking become the focal points of research and technology, researchers begin to realize that the standard layered architecture of the protocol stack, the OSI model does not serve wireless applications well enough. As a result, cross-layer design, which jointly optimizes transmission strategies via enhancing information exchange between different layers, is therefore proposed by many researchers[19, 37].

Researchers from different backgrounds work with different OSI layers. Different types of cross-layer design have been proposed from different perspectives of improving network performance. Fu et al. [20] summarized various cross-layer design methods into three major types including layer-centric, application-specific, and centralized.

1.2.1 Layer-Centric Method

One particular layer (the Application layer or any other layer) is allowed to drive the adaptation of network transmission parameters of other layers by accessing the internal protocol parameters of other layers (usually not adjacent layers) as shown in Figure 1.1(a) and (b). Examples are presented in [11, 17, 30, 40, 41, 43]. Although this method jointly adapts transmission strategies among multiple layers, which in turn greatly improves overall network performance, a couple of drawbacks of this method are listed below.

- Passing information between adjacent layers is not resource intensive, but crossing multiple layers at once requires extra steps to complete. This puts much stress on the energy restricted wireless networks.
  - With this method, cross-layer communications between non-adjacent layers are necessary for optimization. Thus new protocol standards of related layers may be needed to support cross-layer optimization. It is a difficult job to design such a protocol, since the protocol designer cannot just focus on current layer, impacts from other layers also need to be taken care of.

1.2.2 Application-Specific Method

This method provides efficient adaptation of specific applications to the error-prone network [40]. With this method, the lower layers are usually considered as a "black box". The related information (i.e., information about network congestion, packet loss rate) are provided to the application layer so that the application layer can make optimized transmission strategies as shown in Figure 1.1(c). The goal of this solution is to provide applications with necessary information to adpat their transmission

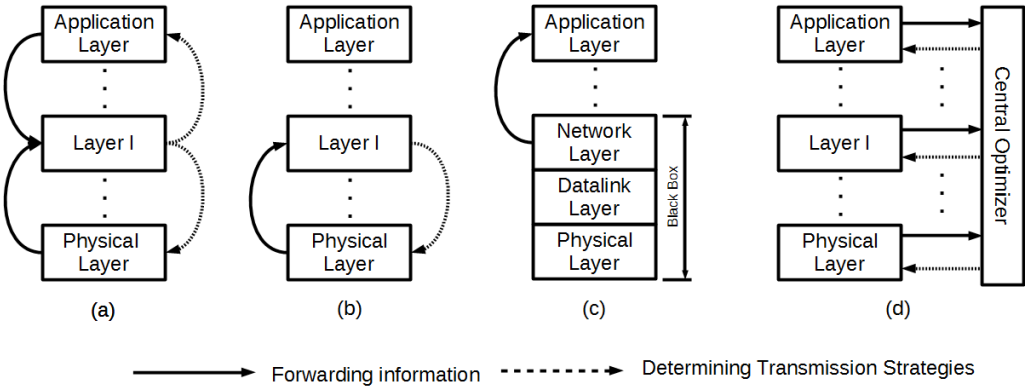


Figure 1.1: Existing cross-layer optimization methods. (a)(b) Layer-centric method. (c) Application-specific method. (d) Centralized method.

parameters without exposing the internal protocol parameters of lower layers. Although the application layer is able to adapt its parameters, this method is not capable of handling adaptations at lower layers.

1.2.3 Centralized Method

This optimization solution uses a centralized optimizer (i.e., a middleware or a system monitor) to dynamically allocate resources across the network based on the observed environmental dynamics and resources of each node in the network as shown in Figure 1.1(d). T. Holliday et al. proposed a centralized optimization system in [22]. The centralized optimizer gathers information, such as channel status, power constraints from the network and wireless nodes, and provides an optimal control policy for each node. This centralized cross-layer optimization requires each layer to forward its protocol-dependent dynamics, as well as its possible protocol parameters to the centralized optimizer, which is a resource extensive process. Each layer also looses the ability to select its protocol parameters locally according to the environmental dynamics it is experiencing. It has to use the parameters determined by the centralized optimizer.

The above mentioned cross-layer design methods optimize protocol parameters by jointly considering environmental dynamics at multiple layers. They also require layers to provide access to some of their internal protocol parameters to other layers. They are resource intensive. They also make independent protocol design for adapting layers impossible without considering the design of related layers. Chapter 2 of this dissertation presents a previous work that utilized the layer-centric cross-layer design to improve network energy efficiency.

In Chapter 3, a new type of cross-layer design, the *autonomous cross-layer optimization*, is introduced to overcome the disadvantages of previously designed cross-layer optimizations. Autonomous cross-layer optimization allows each layer to make their own decision on selecting optimized transmission strategies with only limited information exchange between adjacent layers. Each layer maintains a local optimizer which makes optimized decision on current layer's transmission parameters based upon this layer's environmental dynamics and the limited information from adjacent layer. This cross-layer optimization requires much less information exchange between layers by limiting exchange to adjacent layers.

1.3 Thesis Organization

Chapter 2 presents the design challenges of wireless sensor networks (WSNs), lists a series of effective and energy efficient solutions to WSN problems. A cross-layer optimized energy efficient routing algorithm is introduced and described in detail. Different simulations were performed to demonstrate the higher efficiency of this routing algorithm over another single layer implementation.

Chapter 3 goes deeper into cross-layer design. The advantages of having local optimizers for each layer over optimizations across multiple layers are discussed with

reasoning. Some modifications were made over an existing autonomous cross-layer optimization to make it more computationally efficient and more flexible with various applications. Simulations of the cross-layer optimization algorithms are performed to show the computation speed improvement over the original autonomous cross-layer optimization.

In Chapter 4, ns-3 (a discrete event simulator) simulations of a web server with request queueing are presented. Comparisons among simulations with the original autonomous cross-layer optimization, with the new autonomous cross-layer optimization, and without any cross-layer optimization are done with the quality of service that the application provides. The convergence of the two optimizations are also checked via simulations.

Chapter 5 summarizes the contribution of this thesis. Future works that would make cross-layer design more flexible to applications at runtime and further improve energy efficiency are also suggested.

Chapter 2

Energy Efficient Routing Algorithm for Wireless Sensor Networks

2.1 Introduction

2.1.1 Wireless Sensor Networks (WSN) Design Challenges

WSN technologies are widely used in today's world for various monitoring purposes, such as environmental monitoring[29, 38], medical monitoring[23, 26, 35], security surveillance, home security, etc. In most WSN applications, the network consists of a vast assembly of tiny sensors. These sensors are able to sense the surrounding conditions and transform them into electronic data which is then routed through other sensor nodes back to the sink node as shown in Figure 2.1. The sink node maintains a connection to the server where requests and decisions are made.

Based on the characteristics of WSN, many researchers have pointed out numerous factors that affect the design of the WSN. As suggested in [6] and [7], power consumption, coverage and connectivity, fault tolerance, and scalability are some of the

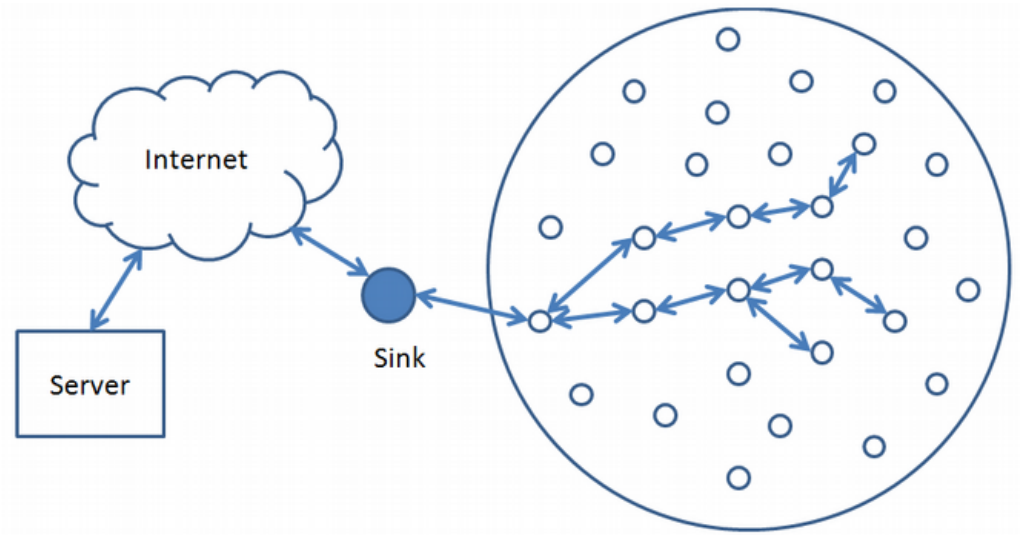


Figure 2.1: A typical WSN.

design factors of WSN.

2.1.1.1 Power Consumption

In large-scale WSNs, such as surveillance applications, thousands of sensors are distributed throughout an area. Once the network is deployed, it is expected to last for a long time (months or years) so as to reduce the initial cost [32]. These sensors are not plugged into power sources. Instead, they usually obtain power from the batteries (or capacitors) that they carry. Due to the small size of the sensors, they do not have much energy capacity. When distributed randomly in the area of interest, it is difficult to locate each sensor in order to maintain it. When sensors run out of energy, it can be impractical to find them and recharge or replace their batteries. Thus energy conservation becomes a major problem in WSN.

2.1.1.2 Coverage and Connectivity

Coverage and connectivity are two fundamental metrics when evaluating the quality and efficiency of a WSN. Each sensor has a sensing range that limits the area that

the sensor can monitor. Information on the area of interest should be as complete as possible, so coverage of the area of interest is another important parameter in WSN.

Connectivity is a third attribute. Once an event is sensed, every bit of detected information must be guaranteed to be reliably transmitted back to a base station. For instance, in a surveillance system, when an intruder is detected, a good coverage ensures that the path of the intruder is tracked. Meanwhile, a higher degree of connectivity assures that detected information gets transmitted to the base station.

2.1.1.3 Fault Tolerance

During data collection, some paths may fail due to the limited transmission power available on sensor nodes, congestion, or physical or architectural structures that interfere with signals. A network should be able to find alternative paths that have more energy or forward less traffic instead of dropping the data. As a result, transmission path redundancy is essential in fault tolerant WSN.

2.1.1.4 Scalability

In many WSN applications, for example, border surveillance and disaster response, there are hundreds and thousands of randomly deployed sensor nodes. The designed network must be able to host a huge number of sensor nodes.

2.1.2 Solutions to WSN Problems

As described above, energy consumption is an essential issue in WSN. Achievements in electronics and wireless communication in recent years made the development of low cost energy efficient WSN possible. There are mainly three different methods

available including designs for: energy efficient devices for sensor nodes, energy saving MAC (Media Access Control) layer schemes, and energy efficient routing protocols for the Network layer.

2.1.2.1 Energy Efficient Wireless Devices

Recent development of Micro-Electro-Mechanical Systems (MEMS) technology makes it possible to design and maintain small, energy efficient devices. PicoRadio [34] and Smart Dust [25] are two examples of lightweight energy efficient sensor devices.When built into the floor of a building, a PicoRadio harvests energy via vibrations. For Smart Dust nodes, the power consumption is limited to roughly 10 microwatts; solar cells could be employed to replenish as much energy as possible when the sun shines or when room lights are turned on.

Both devices support energy efficient communications. For PicoRadio, the sensor data rate is quite low, typically less than one packet/second. A single node's activity duty cycle is typically less than one percent. Thus sensor nodes do not need to be awake all the time. Localization is another feature of PicoRadio, there is no need for the sensors to setup a connection to the destination. The network protocol itself can direct requests for information to the region of interest. This leads to substantial energy savings in radio transmission. In Smart Dust, one can choose between optical and radio frequency (RF) transmission options. Due to the tiny size of Smart Dust, only very limited space is reserved for antennas. Thus using RF transmission, Smart Dust can only provide extremely short-wavelength (i.e., high-frequency) transmission, which is not compatible with low power operation. However, free-space optical transmission requires significantly lower power consumption than the RF counterpart when a line-of-sight path is available. Due to the tiny size of Smart Dust, it is statistically unlikely that any Dust mote might be optically blocked by

another one. Thus passive optical transmission is used by Smart Dust to reduce energy consumption.

2.1.2.2 Energy Saving MAC Layer Scheme

Low power wireless sensor nodes are usually deployed at a high density, due to limited communications range and the need for connectivity. The data rate is quite low for most of the monitoring periods, due to the need to conserve power. To save power, the network is designed so that sensor nodes do not need to be activated at all times. It saves a huge amount of energy when nodes turn off their radios or other communications when they are not transmitting data. Wu et al. [44] proposed an energy efficient "wake up scheduling" for data collection and data aggregation in WSN. Channel access scheme used is time division multiple access (TDMA), within which sensor nodes are scheduled with consecutive time slots for different radio states. Using this scheme, sensor nodes only need to wake up at most twice in one scheduling period, either to transmit or to receive. State transitions are therefore reduced significantly compared to the node-based method in [18]. This in turn saves energy because turning radios on consumes more energy than running them.

2.1.2.3 Energy Efficient Routing Protocols

Researchers developed many different network layer protocols to efficiently route information with low power consumption. Equipped with tiny antennas, the sensors are limited to communicating at high carrier frequency which consumes more power for communicating at the same range at a lower frequency. A good routing protocol reduces collisions, and thus reduces number of retransmissions, which in turn reduces power consumption. A good routing protocol also distributes power consumption among all sensors to maintain longer network lifetime, (i.e., the time for which a

sufficient number of sensors are active to allow the network to function). There are several definitions of network lifetime [12]. In this dissertation, lifetime is defined as the time period between the deployment of the network and when there are not enough sensor nodes remaining to form a network that provides sufficient coverage and connectivity.

The "selective flooding" routing protocol [39] saves energy by selectively flooding routing broadcast packets to appropriate nodes. During the network configuration phase, nodes are grouped into clusters and specific nodes are selected as "cluster heads" that receive data from neighboring nodes. Gateway nodes are used to relay data packets between cluster heads when the distance between two cluster heads is larger than the radio transmission range. Throughout the data transmission phase, these nodes do more communication work than nodes that are not cluster heads. As a result, the cluster heads exhaust their power sooner than non-head nodes, which shortens the total network lifetime.

In the energy efficient routing protocol [8], a directed graph structure is induced upon the nodes by counting hops to a selected sink node; this gives each node possibly multiple parents (with the same lowest hop count to the sink) and perhaps several siblings that are children of the same parent. Each node maintains a routing table which includes a list of parent nodes' information (node ID, hop count and energy level) and sibling nodes' information. It selects its next hop node based upon the energy levels of all the parent nodes and sibling nodes. This routing protocol utilizes cross-layer design (CLD) by allowing the network layer to access a node's energy level at the physical layer directly. However, in this design sensor nodes may relay data packets to their sibling nodes instead of to their parent nodes (when the sibling nodes have more remaining energy than the parent nodes). Thus this algorithm balances power consumption of the network, but doesnot guarantee

minimum transmission latency.

Zeng et al. [45] proposed an energy efficient geographic routing protocol which makes routing decisions locally by jointly considering multiple factors : the observed wireless channel condition, packets advancement to the destination, and the energy availability on each node. This algorithm saves node energy, and guarantees short transmission paths. However each node determines its next hop based upon the above mentioned factors of data transmission at runtime. This introduces a significant amount of both computational and communication overhead at each node. Thus the transmission latency from source to destination is significantly increased.

The energy efficient m-coverage and n-connectivity routing (EECCR) algorithm in [24] tries to balance the energy consumption among all sensor nodes by dividing them into several scheduling sets which are activated periodically to collect and transmit data to the sink node. EECCR has the property that each scheduling set has the ability to cover every location within the region of interest with at least m sensor nodes. Each node has at least n different paths to the sink node. However, since it allows the same node to be included in many scheduling sets without checking the remaining energy of each node, it can dramatically reduce the potential lifetime of certain nodes and thus of the whole network (due to losss of coverage and/or connectivity).

2.1.2.4 Energy Conserving *m*-Coverage and *n*-Connectivity Routing Algorithm

The EECCR routing algorithm proposed in [24] initially divides sensor nodes into disjoint scheduling sets. Nodes in each scheduling set have the property that they together make the monitored region m-covered and the network n-connected, meaning that this routing algorithm has the ability to cover every location within the region

of interest with at least m sensor nodes, and each node has at least n different paths to the sink node. The authors claim that this algorithm balances power consumption among the sensor nodes by allowing the network to switch among using different scheduling sets. However, this algorithm does not consider each node's energy level. Nodes choose neighbors with minimal hop count to the sink to relay information. Therefore the neighbors with minimal hop counts are typically overused. This in turn causes these neighbor nodes exhaust power quickly. The authors also tried to reduce energy consumption on unused nodes by favoring the neighbors that are already chosen by other nodes as their next hop. However, this actually increases data flow through those nodes.

In an attempt to address the issues mentioned above, this chapter introduces a new energy conserving m-coverage and n-connectivity routing algorithm which solves the problems of the EECCR algorithm by including CLD as used in [8]. The following changes are made over the EECCR algorithm:

- The new algorithm monitors sensor nodes' energy levels, and activates only the nodes with sufficient energy to continue functioning, to reduce unnecessary communications between usable nodes and dying nodes.
- When selecting next hop nodes, the probability of a neighbor node being selected is calculated by considering both its hop count and its energy level. Thus not only will power consumption be distributed among all the neighbor nodes, but also packet transmission latency from source to destination will be improved.
- When configuring scheduling set membership for nodes after they get their hop counts via a flooding algorithm as explained in step 2 of routing setup phase in [24], the EECCR algorithm initiates all the nodes with a random scheduling set number. Hence possibly including too many nodes in each scheduling set. In

the new algorithm, only the nodes with highest hop count are initialized with random scheduling set numbers at the beginning of this step; the rest have no initial scheduling set assigned. Using this method, each sensor node eventually belongs to fewer scheduling sets than when using the previous method. Thus there are fewer nodes activated at any time. The network power consumption is reduced, which in turn prolongs the network lifetime.

- Multiple sink nodes are used to further distribute power consumption of nodes and to shorten packet transmission delays.

2.2 The New Energy Conserving *m*-Coverage and *n*-Connectivity Routing (ECR) Algorithm

2.2.1 Coverage and Connectivity Computation

2.2.1.1 m-Coverage Ratio Calculation

The computation method of minimum number of nodes needed to satisfy m-coverage and n-connectivity for the monitored region is derived from the EECCR algorithm in [24] without considering coverage contributions on the edge of the sensing range (the "border effect" described in [24]).

Assume there are N_{Total} sensor nodes randomly distributed in a circular region Ω with radius R. The area of the region is denoted as $||\Omega||$. The expected m-coverage ratio is given by Equation 2.1

$$E[COV_m] = \frac{1}{||\Omega||} \int_{p \in \Omega} \int P(p\, is\, m - covered) d\Omega \tag{2.1}$$

Assume each scheduling set has N nodes. As in [24], suppose that the sensors are spatially distributed according to a Poisson point process with intensity $N/||\Omega||$. Wireless homogeneous networks are considered in this dissertation. Assume sensor nodes have the sensing range of r_s and transmission range of r_t. In this dissertation, border effects are not considered, because sensors are usually overly populated in the case of a large scale network like border surveillance. Thus the calculation of the expected value of m-coverage ratio can be simplified as in Equation 2.2.

$$\begin{aligned} E[COV_m] &= 1 - \tfrac{2}{R^2} \int_0^R (\sum_{x=0}^{m-1} \tfrac{\mu^x}{x!} e^{-\mu}) r dr \\ &= 1 - \sum_{x=0}^{m-1} \tfrac{\mu^x}{x!} e^{-\mu} \end{aligned} \tag{2.2}$$

A location can be covered by all the sensor nodes that are within a range of radius r_s of the location. The average number of nodes in a scheduling set that cover a location is then calculated as $\mu = \frac{N r_s^2}{R^2}$.

2.2.1.2 n-Connectivity Ratio Calculation

As stated in [24], the probability for a region to be n-connected depends on node density and node radio ranges. Let $G(r_t)$ be a directed communication graph that is formed by all nodes whose radio ranges are set to r_t. That $G(r_t)$ is n-connected is a necessary condition for a directed communication graph G to be n-connected, although it is not a sufficient condition. Thus

$$P(G\, is\, n-connected) \geq P[G(r_t)\, is\, n-connected] \tag{2.3}$$

On the other hand, that $d_m[G(r_t)] \geq n$ is a necessary condition for $G(r_t)$ to be n-connected, although it is not a sufficient condition. Here $d_m[G(r_t)]$ denotes the degree of graph $G(r_t)$. Therefore:

$$P[G(r_t)\, is\, n-connected] \leq P\{d_m[G(r_t)] \geq n\} \tag{2.4}$$

In [31], Penrose proved that if N is large enough, with a high probability, if one starts with an empty graph and adds the corresponding links as the radio range increases, the resulting graph becomes n-connected at the moment when it achieves a minimum node degree n, i.e.,

$$P[G(r_t)\, is\, n-connected] = P\{d_m[G(r_t)] \geq n\} \tag{2.5}$$

when $P\{d_m[G(r_t)] \geq n\}$ is almost 1.

According to the definition of network n-connectivity, we have

$$\begin{aligned} P\{d_m[G(r_t)] \geq n\} &= \prod_{\forall u \in G(r_t)} P[d(u) \geq n] \\ &= \prod_{\forall u \in G(r_t)} [1 - exp(-\varphi) \sum_{x=0}^{n-1} \frac{\varphi^x}{x!} \end{aligned} \tag{2.6}$$

where $\varphi = N||\Omega(u, r_t) \cap \Omega||/||\Omega||$ is the expected number of nodes within the radio range of node u in $G(r_t)$. Without considering border effect, $\varphi = \frac{N r_t^2}{R^2}$.

Since $N/||\Omega||$ is a Poisson Point Process, the expected n-connectivity probability

provided by N nodes is calculated as in Equation 2.7.

$$E[CON_n] \geq [1 - Y(\frac{Nr_t^2}{R^2})]^N \tag{2.7}$$

where $Y(k) = e^{-k} \sum_{x=0}^{n-1} \frac{k^x}{x!}$.

The detailed steps involved to form the two formulas Equation 2.2 and Equation 2.7 are provided in Section IV of [24].

Given the m-coverage ratio ε and n-connectivity probability η , the minimum number of nodes needed for each requirement, $N_{COV}(m, \varepsilon)$ and $N_{CON}(n, \eta)$ can be derived via Equation 2.2 and Equation 2.7. Therefore the minimum number of nodes needed for the network to be m-covered with ratio ε and n-connceted with ratio η is then computed as in Equation 2.8.

$$N_{ac}(m, \varepsilon, n, \eta) = max\{N_{COV}(m, \varepsilon), N_{CON}(n, \eta)\} \tag{2.8}$$

However, because of the complexity of Equation 2.2 and Equation 2.7, it is not possible to calculate $N_{COV}(m, \varepsilon)$ and $N_{CON}(n, \eta)$ directly. One must instead derive these values numerically.

As stated in [46], if the transmission range r_t of the sensor node is not smaller than twice its sensing range r_s, full network coverage can guarantee full network connectivity. The network n-connectivity ratio changes when the ratio $\frac{r_t}{r_s}$ changes. By changing r_t while keeping r_s constant, the expected network m-coverage ratio and n-connectivity ratio in a circular region with radius 200m are plotted in Figure 2.2 and Figure 2.3.

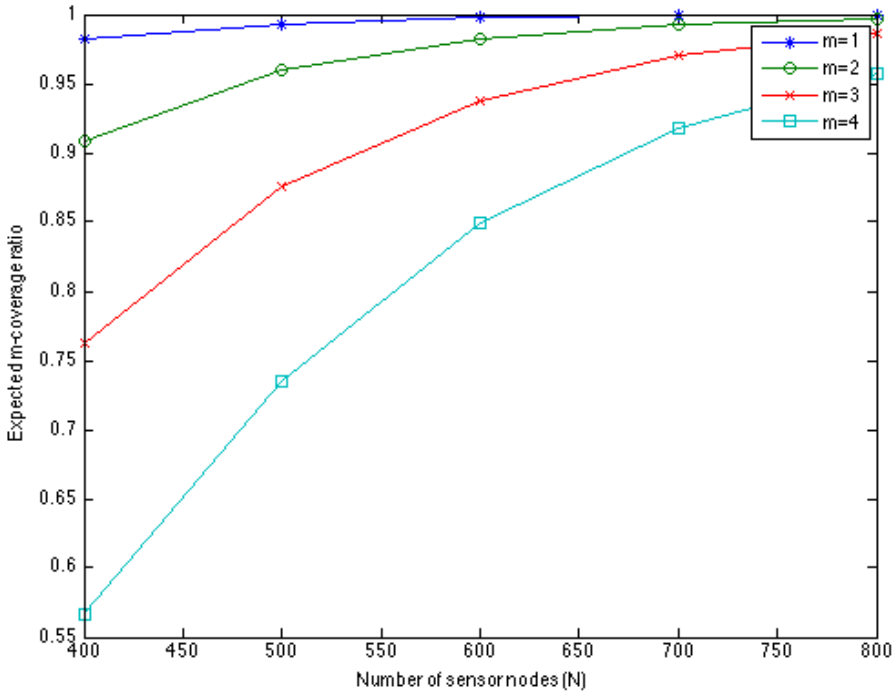
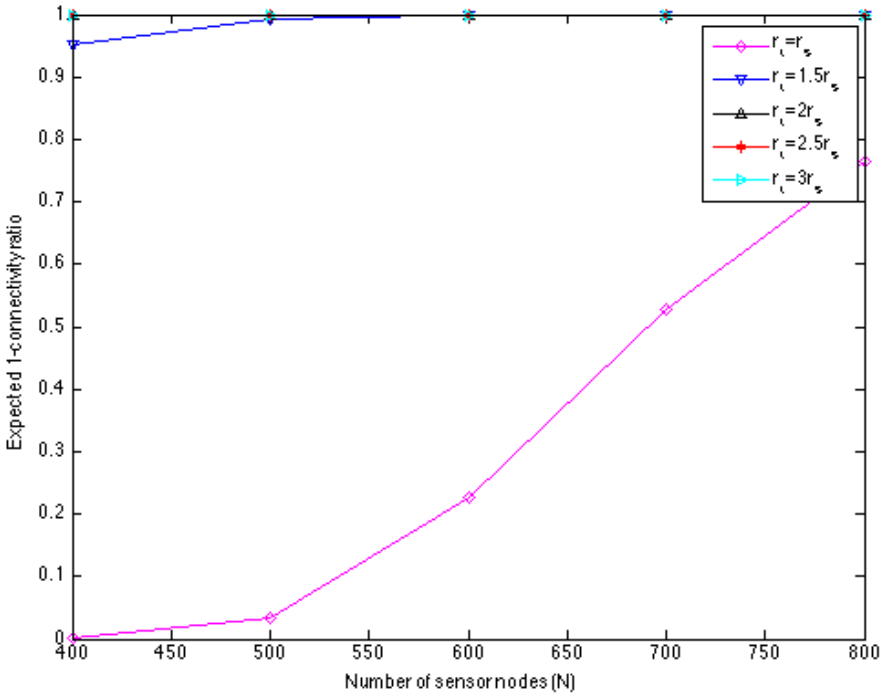


Figure 2.2: Expected m-coverage ratio vs. number of nodes.

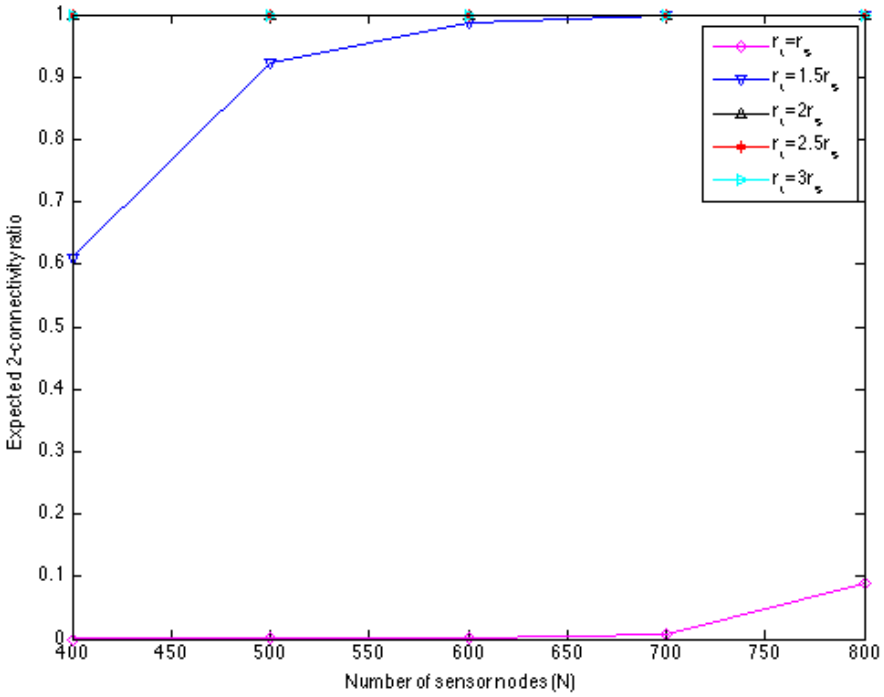
Figure 2.2 gives plots of expected m-coverage ratios with different number of nodes in each scheduling set. Expected *1*-connectivity ratios of different number of nodes with different transmission ranges are also plotted in Figure 2.3. Observed from these plots, when there are 400 nodes in each scheduling set, the network's *1*-coverage ratio is very close to 1. With $\frac{r_t}{r_s} \geq 2$, both the network *1*-connectivity ratio and *2*-connectivity ratio are very close to 1 as long as there are more than 400 nodes available. When $\frac{r_t}{r_s} < 2$, the network connectivity ratios are very low unless thousands of nodes are available.

2.2.2 Energy Model

As mentioned in [14], the sensor nodes consume energy by transmitting and receiving data packets. Wang et al. did research on WSN devices' power consumption



(a) Expected *1*-connectivity ratio vs. number of nodes.



(b) Expected *2*-connectivity ratio vs. number of nodes.

Figure 2.3: Expected n-connectivity ratio vs. number of nodes.

model and proposed a realistic model in [42]. The energy model in this dissertation considers power consumption of nodes running in distinct radio states: sensing, processing, communicating, idle and sleep. Among the active states, communication dominates the energy consumption over the sensing and processing states. Thus in this chapter, for simplicity, only transmission power and receiving power are considered as the energy consumed by the sensor in active states. Power consumed at idle and sleep states are set to constants. Energy consumed by state transition is also taken into account.

If E_{elec} is the energy consumption of transmitting or receiving one bit of data by the transceiver, ε_{amp} is the parameter for the transmit amplifier to achieve the required signal-to-noise ratio, and d is the transmission distance, the energy consumption of transmitting k bits of data is then calculated as in Equation 2.9[14].

$$E_{TX}(k,d) = kE_{elec} + \varepsilon_{amp}kd^2 \tag{2.9}$$

The energy consumption of receiving k bits of data is calculated as in Equation 2.10.

$$E_{RX}(k) = kE_{elec} \tag{2.10}$$

Recall that in the EECCR algorithm, one scheduling set is actively transmitting data at a time. The sensor nodes are in idle listening mode if they are not in an active scheduling set. Sensors consume much less energy in this mode than they are actively sensing and transmitting data packets. When a sensor node has an energy level lower than the threshold value, it goes into sleep mode and is considered dead in the simulation. E_{Idle} and E_{Sleep} are used to represent the energy consumed when

the sensor is in idle or sleep mode. With the WSN under consideration, a node's energy is not replenished. Thus $E_{Sleep} = 0$ in this chapter.

When switching between idle and active states, the sensor node consumes energy for state transition. As in [10], the energy consumed during state transition is modeled as Equation 2.11,

$$E_{StateTransition} = t_{Transition} \times E_{TargetState} \tag{2.11}$$

where $t_{Transition}$ is the time it takes the sensor to transit from one state to another, $E_{TargetState}$ is the power consumption of the sensor node in the target state.

With all the above mentioned energy consumption, in the ECR algorithm, the energy consumed by each sensor node in one scheduling period is modeled by Equation 2.12.

$$E_{Total} = E_{TX}(k,d)+E_{RX}(k)+E_{Idle}(t_{Idle})+E_{Sleep}(t_{Sleep})+E_{StateTransition}(n_{Transitions}) \tag{2.12}$$

2.2.3 Algorithm Description

In this algorithm, data collection contains the routing setup phase and the data transmission phase.

2.2.3.1 Routing Setup

Like in the EECCR algorithm, the routing setup phase is divided into three steps with modifications.

1. In the first step, the maximum number of scheduling sets is calculated the same way as in EECCR algorithm from Equation 2.13,

$$s = \lfloor \frac{N(t)}{N_{ac}(m, \varepsilon, n, \eta)} \rfloor \tag{2.13}$$

where $N(t)$ is the number of available nodes at time t, $N_{ac}(m, \varepsilon, n, \eta)$ is the minimum number of nodes needed for each scheduling set, which is derived from Equation 2.8. Obviously $N(0) = N_{Total}$. If $s < 1$, the network lifetime is considered to be over since there are not enough nodes with sufficient energy to construct even one scheduling set that fullfils m-coverage and n-connectivity ratios for the network. Unlike the EECCR algorithm, the sensor nodes are not assigned a scheduling set number here.

2. Step 2 adopts the method of determining the hop count of each node from the EECCR algorithm, but creates a routing table for each node with additional information in the form of energy level.

   Each node maintains a routing table, in which each item contains a neighbor node's information, including node ID, hop count, and the percentage of available energy. The hop count of the sink node is first initialized to 0. Hop counts of all other nodes are initialized to infinite. The sink node first broadcasts a hello message with the radio range r_t and hop count of 0. The hello message is then propagated throughout the network, increasing the hop count for each hop taken. When a node u receives the hello message from a node v, it sets node v as a neighbor node, and records node v's information as one entry in its routing table. A back-off timer $T_{Backoff}$ is maintained by each node to ensure that the nodes do not receive hello messages from higher hop

count nodes. When the back-off timer of node u expires, node u's hop count is determined as the minimum hop count of its neighbors increased by 1 and no further listening is done. After this, node u broadcasts a hello message with the new hop count, and the next hop nodes set their hop counts and create their routing tables the same way as node u just did.

3. Step 3 constructs the n-connectivity paths for each node in a different way from the EECCR algorithm.

   This step starts with the maximum hop count nodes, assigning each of them a random scheduling set number between 0 and $s-1$ (0 and $s-1$ included). When multiple sink nodes are used, this step starts with assigning local maximum hop count nodes with random scheduling set numbers instead of overall maximum. This is to guarantee that we have enough nodes to start with in multiple sink nodes scenario. After assigning scheduling set numbers, unlike the EECCR algorithm, which uses the neighbor node's hop count as the only factor to select next hop nodes, the ECR algorithm selects next hop node based on both the hop count and the energy level of the neighbor nodes. Each of the sensor nodes (i.e., node u) calculates a probability P of each neighbor node in its routing table for being the next hop node . The P value is calculated by Equation 2.14.

$$P = w_h \times \frac{\triangle HC}{CurrHC} + w_e \times \frac{EL}{E_{Init}} \tag{2.14}$$

where $CurrHC$ is the hop count of current node (i.e., node u). $\triangle HC$ is the difference between the hop counts of current node and its neighbor node. EL is the remaining energy of the neighbor node, E_{Init} is the initial energy for

each node when they were first deployed in the network. w_h ($0 \leq w_h \leq 1$) and w_e ($0 \leq w_e \leq 1$) are the weights given to the two contributing parts, the hop count difference and the energy level.

Node u then selects n(n corresponding to n-connectivity) neighbor nodes with first n biggest values of P as its possible next hop nodes. Then, node u sends notifying messages including its ID and scheduling number set $SN_u(t_1)$ at time t_1 to the n selected next hop nodes. $SN_u(t_1)$ is sent because each node may belong to multiple scheduling sets. If a node v receives a notifying message from node u at time t_2, it updates its scheduling number set to $SN_v(t_2) = SN_v(t_1) \cup SN_u(t_1)$ and repeats the actions that node u has done at time t_1.

It is possible that node u chooses a neighbor node v with the same hop count as its next hop while node v also chooses node u as its next hop node using this method. To avoid this infinite loop, node v is forbidden to select node $u$as its next hop node if node v is already selected by node u as a next hop node.

After these three steps, the network is divided into s scheduling sets with both transmission latency and transmission energy consumption taken care of. These s scheduling sets run in turn in the following data transmission phase.

2.2.3.2 Data Transmission

During the data transmission phase, the scheduling sets are activated periodically to transmit the sensed data to the sink node. Each node (i.e., node u) checks its energy level right after it finishes transmitting in current period. Node u then broadcasts its current energy level with transmission range r_t. Any nodes who have node u in their routing tables as one of their neighbor nodes updates the energy levels of corresponding items in their routing tables.

If in this phase, a node finds that all of its neighbor nodes are having energy levels below a threshold value EL_{thres}, the corresponding scheduling set is eliminated from the list. Here EL_{thres} is determined by E_{Total} calculated in Equation 2.12. Periodically, the network reconstructs new scheduling sets with all the available nodes that have energy levels above EL_{thres}, until there are not enough nodes to make even one scheduling set. The network lifetime is then said to be expired.

2.2.4 Example Networks

A wireless sensor network with a single sink node located in the center of the monitored area, and a wireless sensor network with two sink nodes evenly located in the monitored area are used for demonstrating the energy efficient routing algorithm described in the section above.

Figure 2.4 illustrates how the algorithm works with a single sink node network. The sink node is labeled as "SK", and is placed in the center of the network. Assume after the first step of routing setup phase, we know that the sensor nodes can be divided into 4 scheduling sets. In the second step, each sensor node determines its own hop count according to a flooding algorithm as described in Step 2 of Section 1.2.3.1, and creates a routing table with its neighbor nodes' information. Given region radius R and the transmission range of sensor nodes r_t, the maximum hop count h_{max} for all nodes is no less than R/r_t. Assume $R/r_t = 3$, then nodes with hop count ≥ 3 assign themselves a scheduling set number between 0 and 3 as shown in Figure 2.4(a). Assume $h_{max} = 3$ in this case, at the beginning of the third step, the hop count 3 nodes first start to find their n next hop nodes based on the P values of their neighbor nodes. Assume the application needs to build an m-coverage *1*-connectivity network. Each node only seeks for one next hop node. Upon receiving the notifying messages from hop count 3 nodes, the next hop nodes update their

scheduling number sets as shown in Figure 2.4(b). These next hop nodes do not necessarily have lower hop count than 3, because of the energy level we take into account when selecting next hop nodes. This process goes on until the sink node is reached. A possible result of the routing setup phase is shown in Figure 2.4(d).

Compared to the EECCR algorithm in [24], the ECR algorithm considers both packet transmission latency and nodes' energy levels when selecting the next hop. Thus the ECR algorithm achieves better energy efficiency while maintaining short transmission latency. Figure 2.5 shows an example network with multiple sink nodes. It is easily deduced that with multiple sink nodes, the transmission latency is even smaller than using a single sink node in the network.

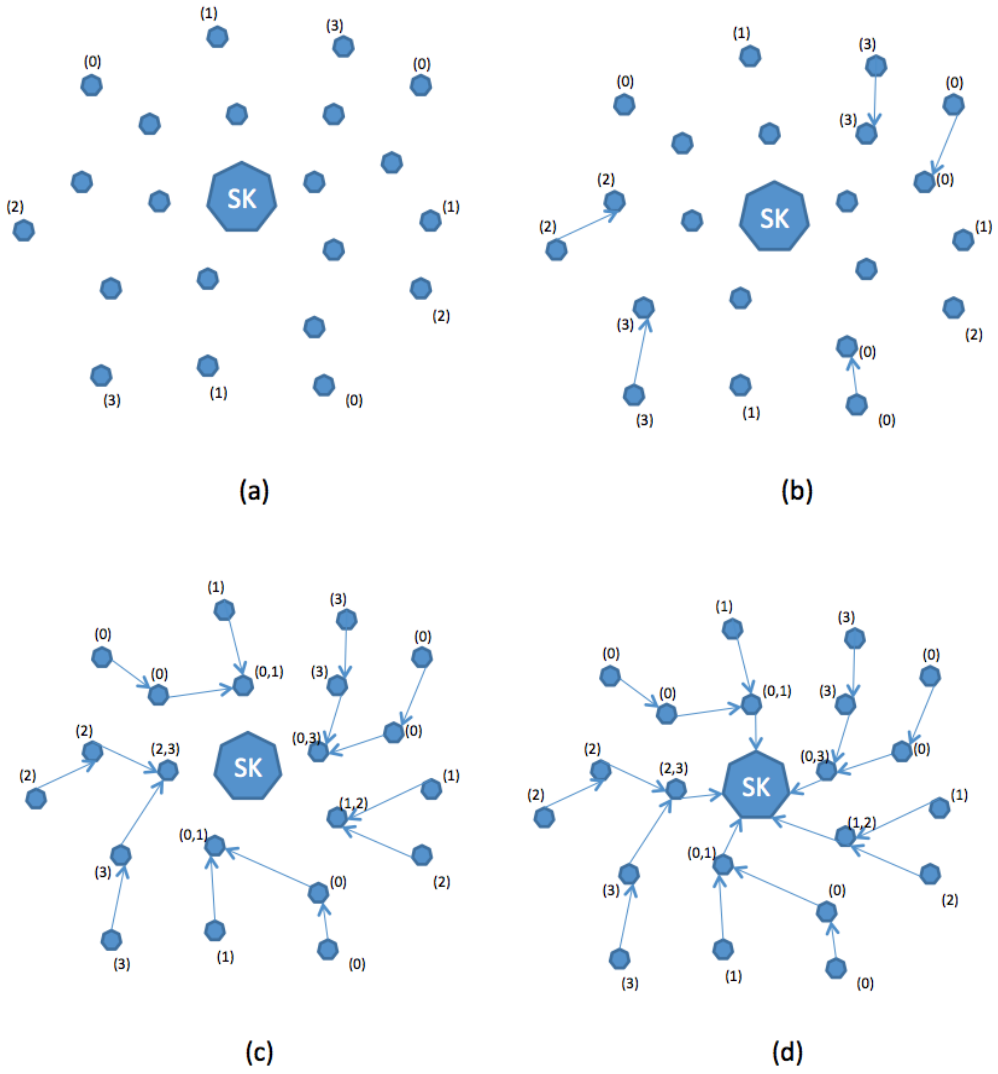


Figure 2.4: Example network with single sink node.

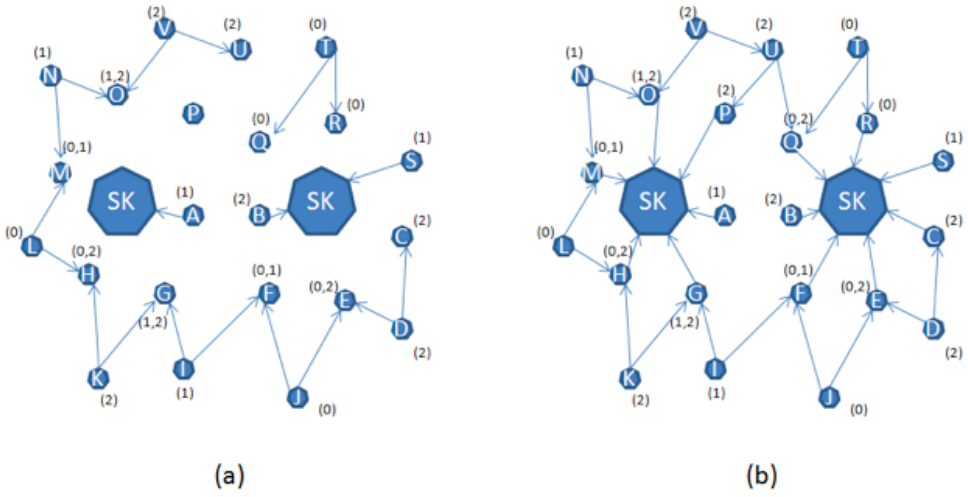


Figure 2.5: Example network with 2 sink nodes.

2.3 Simulation and Results

2.3.1 Simulation Setup

Simulations were conducted in a custom simulator written in Microsoft Visual Studio using C#. Comparisons are done between the EECCR algorithm and the ECR algorithm introduced in this chapter.

In this simulator,

1. all nodes are battery powered with no ability to recharge.
2. once a node sleeps, it is permanently deactivated, so E_{Sleep} does not matter and is set to 0.
3. nodes in idle mode do not consume much energy, so E_{Idle} is set to 0 for simplicity.

Below are the metrics considered during simulations.

1. Network m-coverage ratio v.s. number of active nodes. The simulation loops over the coordinates of the whole monitored region, searches in each circular area with radius r_s, centered at each location. It checks the number of active nodes in that area for each scheduling set. Number of m-covered locations is increased by 1 if there are m active nodes in this circular area. Then it calculates the average percentage of locations that are m-covered in the whole region among all the scheduling sets. This average percentage is the network m-coverage ratio. Given the m-coverage ratio and n-connectivity ratio, the number of nodes required for each scheduling set is determined as in Section 1.2.1. The network m-coverage ratios were calculated with different number of nodes per scheduling set. The results are compared between the EECCR algorithm and the new algorithm.

2. Standard deviation of energy consumption over same hop count nodes. The energy levels of all nodes are checked periodically and the standard deviation of the energy consumption among all nodes with the same hop count is calculated. The values of the standard deviation shows the algorithm's ability to distribute power consumption among nodes with the same hop count. The difference among all standard deviation values of different hop counts demonstrates the algorithm's efficiency of distributing power consumption among all sensor nodes.

3. Network lifetime. Here the network lifetime is defined as the time from the sensor network is first established until there are not sufficient available sensor nodes to achieve the m-coverage and n-connectivity network.

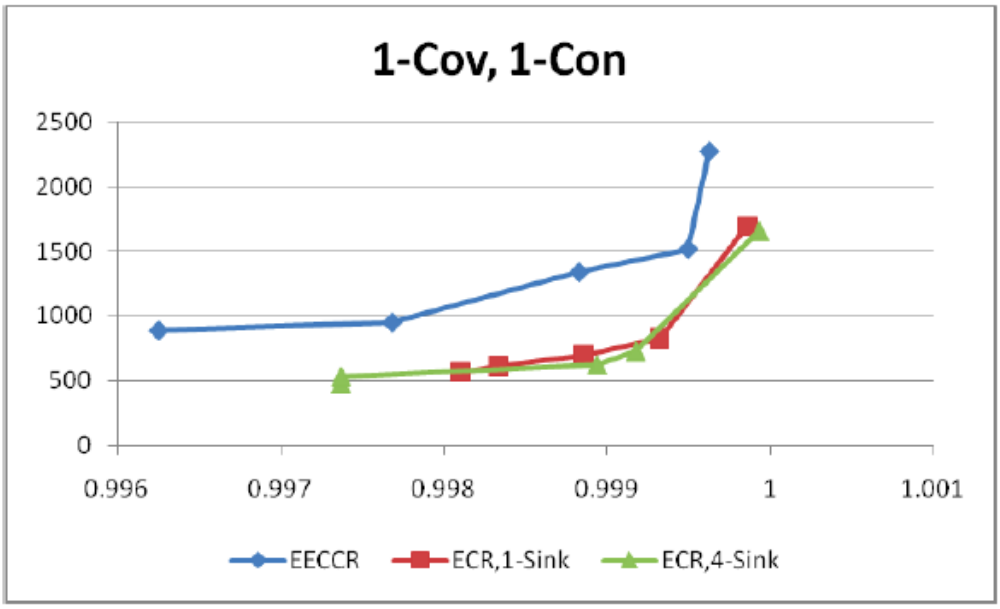
In order to use the simulation results from [24] and compare it with the results of the ECR algorithm, the same simulation parameters were used in these simulations. The simulation parameters are listed in Table 3.2.

Table 2.1: Simulation parameters.

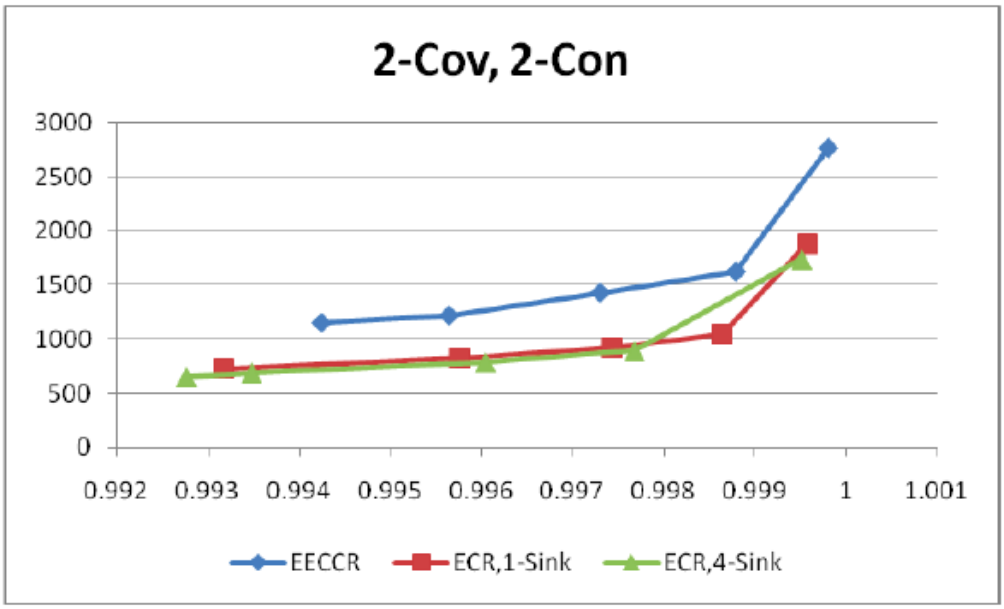
Simulation Parameter	Value
Radius of circular region (R)	200m
Sensing range of sensor node (r_s)	20m
Number of sensor nodes (N_{Total})	5000
System back-off timer ($T_{Backoff}$)	20ms
Length of data transmission phase (T_D)	500ms
Sensor node's initial energy	100
Transmitting energy per packet	2
Receiving energy per packet	1

For simplicity, the nodes' transmission range is set to 40m, which is twice the sensing range. This means network coverage guarantees network connectivity. 5000 homogeneous sensor nodes are randomly distributed in the circular region. Both E_{Idle} and E_{Sleep} are set to 0 for simplicity. Each sensor node generates its first packet at a random time that is at least 20 seconds after the program starts to run. After that it generates packets at random times that are at least 2 minutes apart. Here the random packet generation time is generated by a pseudorandom number generator with uniform distribution. The scheduling set switching interval is set to 10 seconds. The interval between each recalculation of new scheduling sets is set to one minute. Standard deviations of energy consumption are checked every one minute. The ECR algorithm was simulated with single sink node and 4 sink nodes separately. The results are compared with EECCR algorithm in Figure 2.6, Figure 2.7 and Figure 2.8.

2.3.2 Simulation Results and Comparison



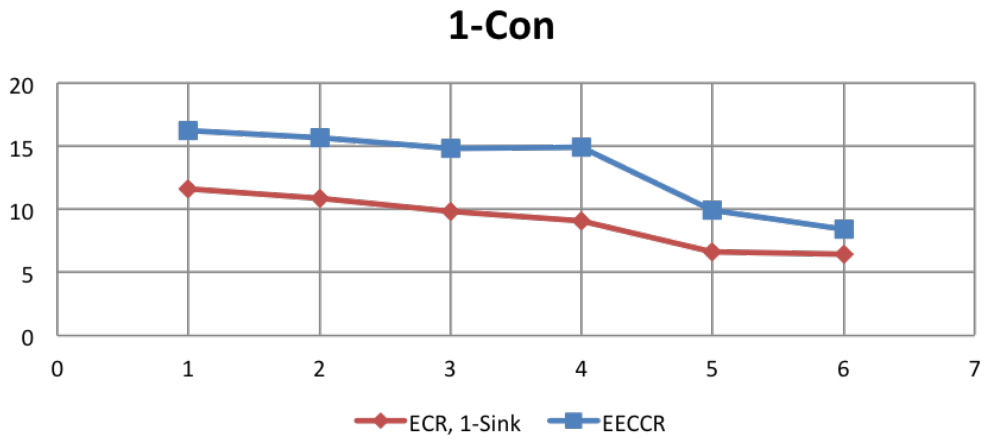
(a) *1*-coverage, *1*-connectivity.



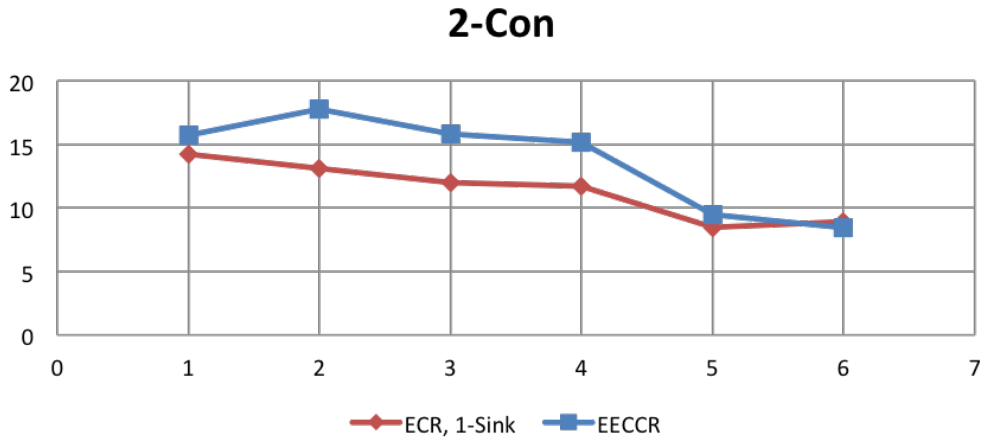
(b) *2*-coverage, *2*-connectivity.

Figure 2.6: Number of active nodes vs. coverage ratio.

Multiple simulations were performed with maximum number of scheduling sets set to 6, 8, 10, 12 and 14. The results of average number of active nodes in the network versus network m-coverage ratio are plotted in Figure 2.6 for both the EECCR algorithm and the ECR algorithm with single sink node and 4 sink nodes. Both plots demonstrate that to achieve the same m-coverage ratio, the ECR algorithm on average requires fewer nodes to be active in the network. This is because in the ECR algorithm, the sensor nodes are included in fewer scheduling sets. Thus there are fewer nodes running at any time.

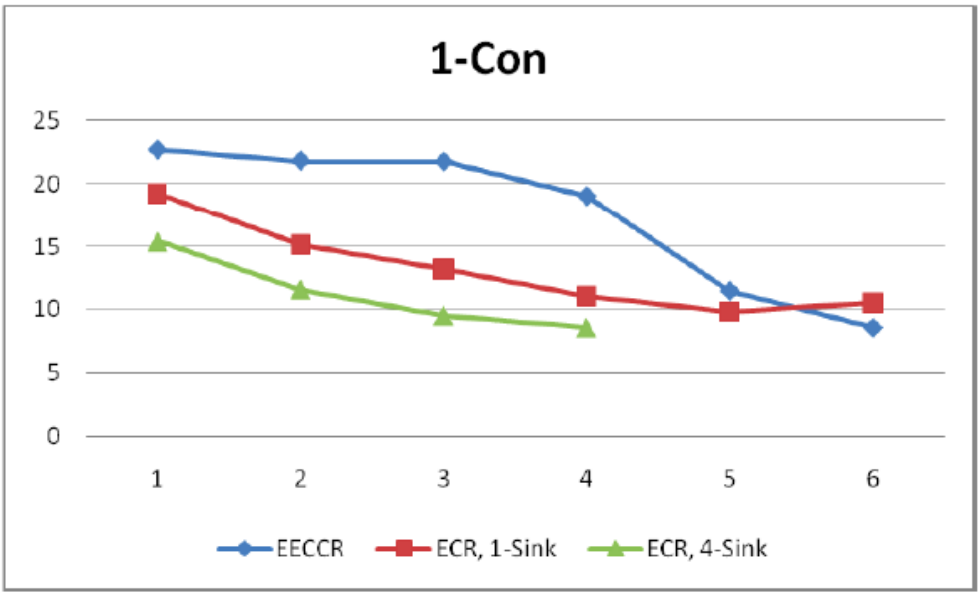


(a) *1*-connectivity.

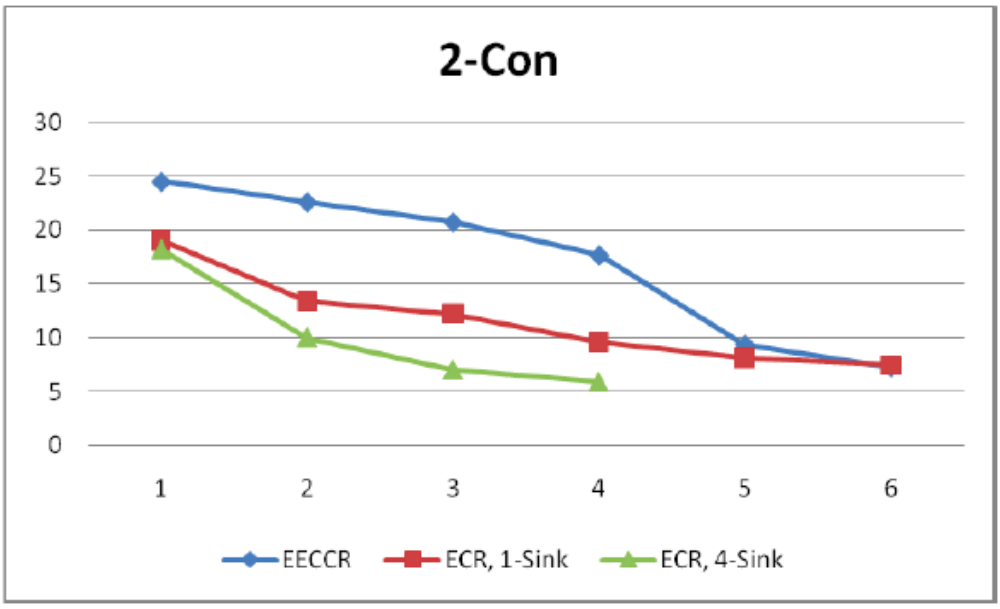


(b) *2*-connectivity.

Figure 2.7: Node usage vs. hop count, $w_e = 1, w_h = 0$.



(a) *1*-connectivity.



(b) *2*-connectivity.

Figure 2.8: Nodes usage vs. hop count, $w_e = w_h = 0.5$.

The standard deviations of energy consumption among the nodes with the same hop count are plotted in Figure 2.7 and Figure 2.8 for all the hop counts. Simulations were performed with maximum number of scheduling sets equal to 6. When the

weight for energy level– w_e – and the weight for hop count– w_h – used are set as $w_e = 1$ and $w_h = 0$, the ECR algorithm is actually a pure energy saving algorithm. The standard deviation of energy consumed among all nodes with same hop count never exceeds the EECCR algorithm. When the weights are set to the same, $w_e = w_h = 0.5$, the plots in Figure 2.8 show that the standard deviation values of the ECR algorithm are still smaller than the EECCR algorithm for most hop counts. These plots indicate that the ECR algorithm tends to spread the power consumption among the nodes with the same hop count better than the EECCR algorithm. The standard deviation values of energy consumption among nodes with hop counts 2, 3 and 4 are much smaller with the ECR algorithm comparing to higher hop counts. This is because when selecting next hop nodes for the higher hop count nodes, energy level of their neighbors are considered, so power consumption among these lower hop count nodes are better distributed. The ECR algorithm with 4 sink nodes has even smaller standard deviation values, which indicates that with 4 sink nodes, the power consumption is further distributed among all nodes. Note that with more sink nodes, the largest hop count is smaller than 1 sink node scenario, because it takes fewer hop counts for the information to reach the sink nodes.

To check if the ECR algorithm really prolongs the network lifetime, more simulations are done by assigning each sensor node with a constant initial energy level, and checking their energy levels regularly during simulation. With an initial energy level of 120, for simplicity, assume packet transmission power and sensing power of each node are both 2, packet receiving power and processing power are both 1, assume number of scheduling sets is 6. When using 50% weights for both the energy level and hop count, the *2*-connectivity network using the ECR algorithm ran 249 seconds with 1 sink node, 283 seconds with 4 sink nodes. While the *2*-connectivity network using the EECCR algorithm only ran for 174 seconds. That means that the

ECR algorithm has more than 40% improvement on energy saving over the EECCR algorithm. Because of the concentrated node usage of the EECCR algorithm, some areas die sooner which causes the shorter network lifetime.

2.4 Conclusion and Future Work

In many large scale wireless sensor networks, small sensors are randomly distributed in great volume which makes battery recharging or replacement impossible. Energy conservation becomes a critical solution to prolonging network lifetime. The EECCR algorithm in [24] divides the whole network to s scheduling sets and lets different sets work alternatively to distribute power consumption among nodes. However, when setting up the scheduling sets, the EECCR algorithm did not take into account nodes' energy level which may cause some nodes to deplete their energy very quickly. In this chapter, a new energy aware routing algorithm, the ECR algorithm distributes data traffic among sensor nodes more evenly by considering both the hop count and the energy level of each node's neighbors while setting up the scheduling sets. This new algorithm utilizes CLD, which allows the network layer to get a sensor node's energy level directly. Simulation results verified that with CLD, the new algorithm prolongs network lifetime compared to the EECCR algorithm while maintaining better network m-coverage and n-connectivity ratios. With multiple sink nodes, the network power consumption was further improved. The transmission latency is also shortened because of the smaller average hop counts between sensor nodes and the sink node. However, the lower hop count nodes still carry most data traffic. These nodes will be the first nodes that deplete their energy. In order to further distribute power consumption, alternative sink nodes located at different locations in the circular region could be activated periodically to change

nodes' hop counts from time to time, which in turn redistributes work load among the sensor nodes periodically.

Chapter 3

The Autonomous Cross-Layer Optimization

3.1 Introduction

The autonomous cross-layer optimization uses individual optimizers for all the OSI layers. Information are accumulated from lower layers and passed to upper layers, and vice versa. Thus each layer only communicates with its neighbors. This provides two advantages over the common cross-layer designs (CLD) that perform optimization based on information from multiple layers apart:

- Crossing multiple layers at once requires extra steps to complete. Whereas passing information between adjacent layers is less resource intensive.
- If communication between non-adjacent layers is necessary, new protocol standards of related layers may be needed for cross-layer optimization to work. Whereas the autonomous cross-layer optimization is compliant with existing standards.

This dissertation did a thorough study of the results from the paper "A New Systematic Framework for Autonomous Cross-Layer Optimization" [20], and made several modifications to improve the application flexibility and computation consumption of the optimization. The following sections explain the autonomous CLD architecture in detail, and give an example of optimization with a Wi-Fi network. The entity relationship diagram in Figure 3.1 is employed in the example.

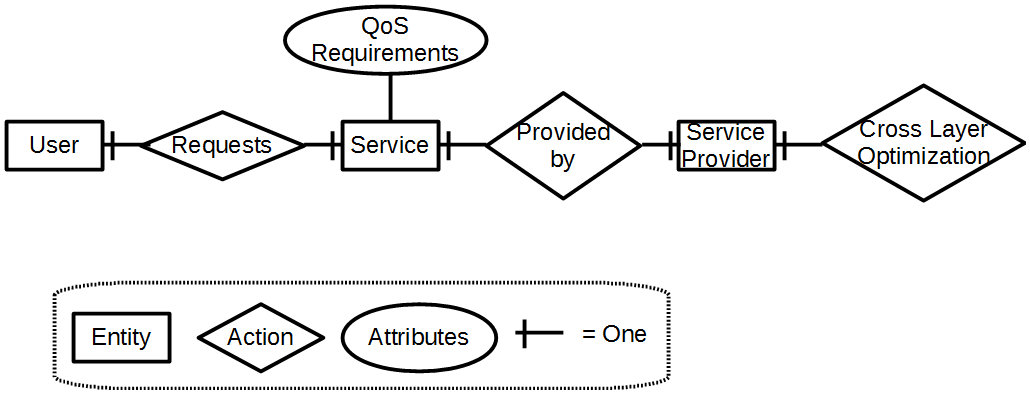


Figure 3.1: Entity relationship diagram of the system.

3.2 Autonomous Cross-Layer Design Architecture

The autonomous CLD [20] modeled as a Markov Decision Process (MDP) makes foresighted transmission strategy adaptation that maximizes the discounted cumulative network utility as in [22, 15, 21].

In this dissertation, a multimedia Wi-Fi network with a one-hop transmission between the Access Point (AP) and a wireless station is optimized. The way the transmitter adapts its transmission strategies at the Physical (PHY), Media Access Control (MAC) layer, and Application (APP) layers to maximize its network utility is carefully described.

Each participating layer is indexed by l where $l \in \{1, \ldots, L\}$. Layer 1 represents the

lowest layer in the stack, the PHY layer, while layer L corresponds to the highest layer, the APP layer. Figure 3.2 shows the autonomous CLD model.

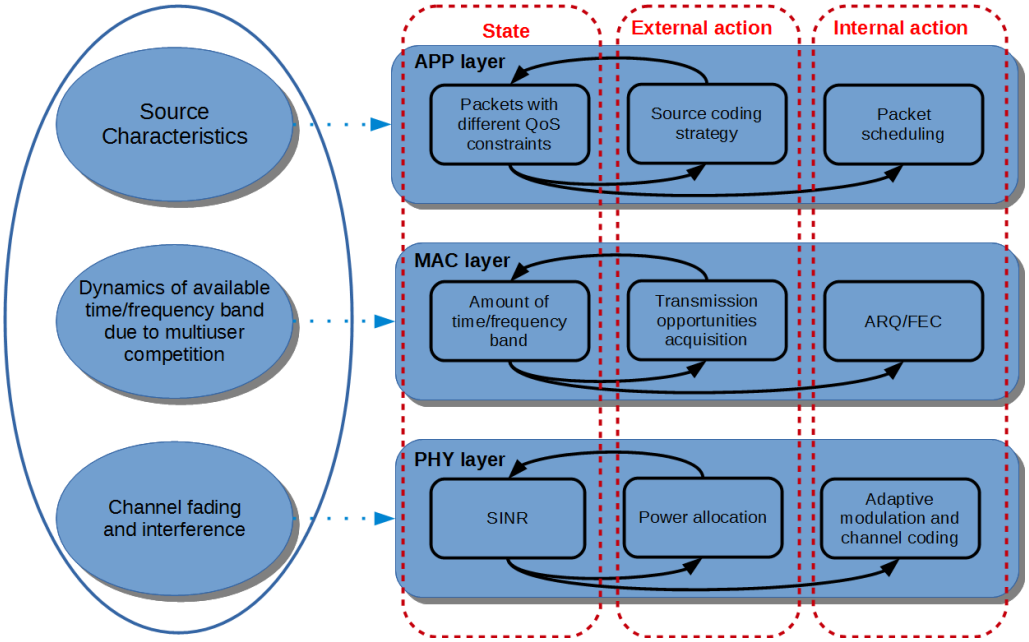


Figure 3.2: Autonomous CLD model.

3.2.1 Basic Concepts

3.2.1.1 Transmission Strategies

The transmission strategies of each layer are defined as follows:

At the PHY layer, data signals are received along with channel noise and interference from other users. The channel quality experienced during the transmission is represented by the Signal-to-Interference-plus-Noise Ratio (SINR). The transmitter's power allocation and modulation scheme are considered as the PHY layer transmission strategies. By adapting power allocation, the received SINR changes. The modulation scheme determines the Quality of Service (QoS) the PHY layer provides to its upper layer.

At the MAC layer, channel access can be based on Frequency-Division Multiple Access (FDMA), Time-Division Multiple Access (TDMA), or Code-Division Multiple Access (CDMA), etc. An advanced form of FDMA is Orthogonal Frequency-Division Multiple Access (OFDMA), in which service providers are assigned different OFDM sub-channels. Transmission opportunity acquisition determines the percent of time that a wireless service provider is assigned with channel access. With TDMA, wireless service provider has to bid for the channel to be used. However, with OFDMA or CDMA, wireless service provider gets channel access all the time. The error-control method Automatic Repeat reQuest (ARQ), or Forward Error Correction (FEC) determines the QoS that the MAC layer provides to its upper layer.

At the APP layer, changing the transmission strategy, the source coding algorithm, changes the amount of packets to be sent at each step.

3.2.1.2 States

As described above, each participating layer is indexed by l where $l \in \{1, \ldots, L\}$. The current state and the next state of each layer are denoted by s_l and s_l' , where $s_l, s_l' \in S_l$. S_l is the set of possible states at layer l. The state of the wireless sender, which is the input state to the CLD algorithm, is then denoted as $\boldsymbol{s} = [s_1, \ldots, s_L] \in \boldsymbol{S}$, where $\boldsymbol{S} = S_1 \times \ldots \times S_L$. The state transitions of the three participating layers are modeled as Finite State Markov Chains (FSMC) as illustrated in [28, 9]. With FSMC, the state transition is a memoryless procedure, meaning that the transition from current state to next state does not depend on the history of state precedes the current state. Therefore, given the current state, future transmission strategies are determined independently of the past history of the transmission strategies and environment.

This Markovian model is an approximation of a non-Markovian real entity, since

in reality, nothing is actually memoryless. Based on the observed parameters, it is okay to assume a Markovian model for convenient approximation. In reality, a non-Markovian system with memory slows down the process, since the previous memory has impact on the process. However, with caching, a non-Markovian process might be running faster than the Markovian counterpart.

3.2.1.3 Actions

Two types of transmission actions –the external action and the internal action– are defined to determine the transmission strategies that are used at each layer. The transmission parameters set by the external action at layer l determines the next state of that layer, whereas with an internal action, the corresponding transmission strategy in layer l, combined with the service provided from lower layer determines the QoS at current layer.

The output state of the external action at layer l is denoted by $a_l \in A_l$, where A_l is the set of the possible external actions can be taken at layer l. The output state of the internal action at layer l is denoted by $b_l \in B_l$, where B_l is the set of possible internal action output states available at layer l. The aggregation of external action and internal action output states, denoted by $\xi_l = (a_l, b_l)$ represents the output state of action at layer l. Each of these combined actions sets a pair of output states in the optimization algorithm, based upon observed input states.

3.2.1.4 Transition Probability

Since the states at each layer are Markovian, the state transition only depends on the current input state, the current performed external action, and the observed environmental dynamics, as embodied by input states. The corresponding state transition probability is denoted by $p\left(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{\xi}\right)$. In the cross-layer optimization model

considered in this chapter, the state transition at each layer $l < L$ is only controlled by the external actions at that layer and is independent of other layers' states and actions. At layer L, the state transition is determined by the external actions at that layer and internal actions of all the layers. Motivated by this model, the transition probability for the cross-layer optimization is simplified as:

$$p\left(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{\xi}\right)=\prod_{l=1}^{L-1} p\left(\boldsymbol{s}_l'|s_l,a_l\right)p\left(s_L'|s_L,a_L,\boldsymbol{b}\right) \tag{3.1}$$

3.2.1.5 System Reward

The application quality $g\left(\boldsymbol{s},\boldsymbol{b}\right)$ at layer L is determined by the states and internal actions at each layer. Performing the internal actions at various layers will incur the internal cost $d(s,\boldsymbol{b})$, which will be set to zero if no cost is incurred. The external cost $c_l(s_l,a_l)$ at layer l represents the cost of performing the external action, e.g., the amount of power allocated to determine the channel conditions or the tax (tokens, money) spent for consuming wireless resources. Thus the system reward is defined as:

$$R\left(\boldsymbol{s},\boldsymbol{\xi}\right)=g\left(\boldsymbol{s},\boldsymbol{b}\right)-\lambda^b d\left(\boldsymbol{s},\boldsymbol{b}\right)-\sum_{l=1}^{L}\lambda_l^a c_l\left(s_l,a_l\right) \tag{3.2}$$

where λ^b is a positive parameter that weighs the internal cost $d(\boldsymbol{s},\boldsymbol{b})$ of performing internal actions $\boldsymbol{b}$ into the system reward. The internal cost is aggregated to the top layer because the internal action taken at each layer along with the service provided by lower layer determines the QoS of the current layer. λ_l^a is a positive parameter that weighs the external cost $c_l(s_l,a_l)$ of performing the external action a_l at layer

l into the system reward. These parameters are determined by the application administrator, based on the available resources of the service provider. The more resources available, the lower the cost should be.

Since actions are distinguished as internal and external, the system reward can be decomposed into two parts: the internal reward, which depends on the internal actions; and the external reward, which depends on the external actions. The internal reward is:

$$R_{in}(\boldsymbol{s}, \boldsymbol{b}) = g(\boldsymbol{s}, \boldsymbol{b}) - \lambda^b d(\boldsymbol{s}, \boldsymbol{b}) \tag{3.3}$$

while the external reward is:

$$R_{ex}(\boldsymbol{s}, \boldsymbol{a}) = -\sum_{l=1}^{L} \lambda_l^a c_l(s_l, a_l) \tag{3.4}$$

Thus $R = R_{in} + R_{ex}$.

3.2.1.6 Quality of Service

The internal action combined with the QoS in a lower layer determines the QoS it provides to its upper layer. In the autonomous CLD algorithm, QoS in each layer, denoted by $Z_l, l \in [1, \ldots, L]$, is defined with three elements:

- the packet loss probablity ε_l, which presents the probability that one packet is lost at layer l during transmission;
- the packet transmission delay τ_l at layer l;
- and the packet transmission cost υ_l at layer l.

Thus $Z_l = (\varepsilon_l, \tau_l, \upsilon_l)$. The relationship between QoS of adjacent layers is denoted by the function $Z_l = \overrightarrow{f_l}(s_l, b_l, Z_{l-1})$. Hence to calculate the internal reward, layer L needs to know all the QoS levels jointly determined by the states and internal actions at all layers. Given the current state $\boldsymbol{s}$, the set of possible QoS levels at layer l is

$$\boldsymbol{\mathcal{Z}}_l(s) = \left\{ \begin{array}{rcl} Z_l | Z_l & = & \overrightarrow{f_l}(s_l, b_l, Z_{l-1}), \ldots, \\ Z_1 & = & \overrightarrow{f_1}(s_1, b_1, \varnothing) \ \ \forall b_1 \in B_1, \ldots, b_l \in B_l \end{array} \right\}. \tag{3.5}$$

This QoS set is then provided to the upper layer for its computation of QoS set $\boldsymbol{\mathcal{Z}}_{l+1}(\boldsymbol{s})$. However, the size of $\boldsymbol{\mathcal{Z}}_l(\boldsymbol{s})$ is often very large, which leads to heavy computation at the upper layer. Hence, instead of providing the complete set of $\boldsymbol{\mathcal{Z}}_l(\boldsymbol{s})$, a reduced sized version, called the frontier of the set, is provided to the upper layer.

For two QoS levels, $\mathcal{Z}_l = (\varepsilon_l, \tau_l, \upsilon_l)$ and $\mathcal{Z}'_l = (\varepsilon'_l, \tau'_l, \upsilon'_l)$, $\mathcal{Z}'_l$ is said to dominate $\mathcal{Z}_l$, denoted by $\mathcal{Z}'_l \overset{d}{\leq} \mathcal{Z}_l$, if any of the three elements in $\mathcal{Z}'_l$ is smaller than that of $\mathcal{Z}_l$. The algorithm that computes the QoS frontier at layer l is presented in Figure 3.3.

3.2.2 The Foresighted Cross-Layer Optimization

In this dissertation, the state transition is discretized into stages, which is denoted by the letter k as a superscript, where $k \in \mathbb{N}$. At the beginning of each stage, the CLD algorithm does optimization for the transmission. It selects the optimal internal and external actions at each layer that maximize the system reward. During the stage, the service provider has constant state and performs the same actions. The length of each stage is denoted by $\triangle T$, which is also the length of each optimization cycle. Then the state of the service provider at stage k is denoted by $\boldsymbol{s}^k$, with s_l^k being the

Input: $\mathcal{Z}_{l-1}$, s_l, and B_l.
Initialize: $\mathcal{Z}_l = \varnothing$, flag $= 0$.
Loop 1: For each $b_l \in B_l$
Loop 2: For each $Z_{l-1} \in \mathcal{Z}_{l-1}$
flag $= 0$;
Compute $Z_l = \vec{f}_l(s_l, b_l, Z_{l-1})$.
Loop 3: For each $Z'_l \in \mathcal{Z}_l$
If $Z'_l \overset{d}{\leq} Z_l$
flag $= 1$; break;
endif
endfor //loop 3
if flag == 0
$\mathcal{Z}_l = \mathcal{Z}_l \cup \{Z_l\}$.
endif
endfor //loop 2
endfor // loop 1

Figure 3.3: Algorithm for constructing QoS frontier $\mathcal{Z}_l$.

state of layer l at that stage. The joint action at stage k is now denoted by ξ^k, with $\xi_l^k = \left(a_l^k, b_l^k\right)$.

The foresighted cross-layer optimization aims to find the optimal internal and external actions that maximize the discounted cumulative reward. So not only is the immediate reward considered, but also the selected actions' impact on future reward is taken into account. However, the immediate reward should be given a higher weight since the transmission channel makes unexpected changes constantly. The discounted cumulative reward is defined as described in [22, 15, 21].

$$\sum_{k=0}^{\infty} \gamma^k R\left(\boldsymbol{s}^k, \boldsymbol{\xi}^k | \boldsymbol{s}^0\right) \tag{3.6}$$

where γ is the discounted rate with $0 \leq \gamma < 1$. $\boldsymbol{s}^0$ is the initial state in which the application initially started transmission.

The foresighted cross-layer optimization is formulated using a Markov Decision Process (MDP). The MDP is defined as follows.

Definition 1: MDP is defined in [33]as a tuple $M = (\mathcal{S}, \mathcal{X}, p, R, \gamma)$, where $\mathcal{S}$ is a joint state space, $\mathcal{X}$ is a joint action space for each state, p is a transition probability function $\mathcal{S} \times \mathcal{X} \times \mathcal{S} \mapsto [0, 1]$, R is a reward function $\mathcal{S} \times \mathcal{X} \mapsto \mathfrak{R}$, and γ is the discounted factor.

In this dissertation, the joint state space is $\mathcal{S} = \mathcal{S}_1 \times \ldots \times \mathcal{S}_L$, the joint action space is $\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_L$, the transition probability is given by Equation 3.1, and the reward function is given by Equation 3.2.

Optimization of MDP can be accomplished via either linear programming or dynamic programming. Linear programming is typically used to solve continuous-time MDP while dynamic programming is typically used to solve discrete-time MDP [4].

With linear programming (LP), optimization can be done at any time. However, the advantage of taking optimal actions only benefits the system when the current state changes. It is not advantageous for our algorithm to use linear programming since the state transition is discretized into stages.

Dynamic programming (DP) has the advantage of splitting a complicated problem into subproblems. Value-iteration or policy-iteration is usually used to do dynamic programming. In this dissertation, value-iteration is used to perform dynamic programming stage by stage to solve the optimization problem. In each stage, it uses the same calculation, which reduces computation complexity.

In the next chapter, ns-3, a discrete event network simulator is used to do further simulations with the autonomous CLD. This also makes dynamic programming a preferred choice for optimization.

The cross-layer optimization at each layer is called the DP operator. The key step

performed at each iteration is to solve the following optimization problem.

$$\max_{\xi \in \mathcal{X}} \left\{ R\left(\boldsymbol{s}, \boldsymbol{\xi}\right) + \gamma \sum_{s' \in \mathcal{S}} p\left(\boldsymbol{s}' | \boldsymbol{s}, \boldsymbol{\xi}\right) V\left(\boldsymbol{s}'\right) \right\} \tag{3.7}$$

where $V\left(\boldsymbol{s}'\right)$ is a state value function that is defined as the discounted reward when starting from state $\boldsymbol{s}'$.

Substituting the two parts in Equation 3.7, the DP operator can be rewritten as follows:

$$\max_{\boldsymbol{a} \in \boldsymbol{A}, \boldsymbol{b} \in \boldsymbol{B}} \left\{ \begin{array}{l} \underbrace{g\left(\boldsymbol{s}, \boldsymbol{b}\right) - \lambda^b d\left(\boldsymbol{s}, \boldsymbol{b}\right) - \sum_{l=1}^{L} \lambda_l^a c_l\left(s_l, a_l\right)}_{R(\boldsymbol{s}, \boldsymbol{\xi})} \\ + \underbrace{\gamma \sum_{s_1' \in \mathcal{S}_1, \ldots, s_L' \in \mathcal{S}_L} p\left(s_1' | s_1, a_1\right) \cdots p\left(s_L' | \boldsymbol{s}, \boldsymbol{b}, a_L\right) V\left(s_1', \ldots, s_L'\right)}_{\sum_{\boldsymbol{s}' \in \boldsymbol{S}} p(\boldsymbol{s}' | \boldsymbol{s}, \boldsymbol{\xi}) V(\boldsymbol{s}')} \end{array} \right\} \tag{3.8}$$

From Equation 3.8, the DP operators at the three layer considered in this dissertation are derived in Table 3.1. In this table, R_{in} is the internal reward obtained by the system, Z_3 is the QoS received by the APP layer. The fact that the impact of taking internal actions at lower layers is aggregated to the QoS received at the APP layer, and that there is no internal action taking at the APP layer, makes the internal cost $d(s, \boldsymbol{b}) = 0$. c_1, c_2, c_3 are the costs generated by the external actions taken at the three layers. In this cross-layer optimization model, we use the values of external actions at each layer as the cost incurred by taking these external actions. $\lambda_1^a, \lambda_2^a, \lambda_3^a$ are the positive parameters that trade off between the application quality and the costs incurred by taking external actions a_1, a_2 and a_3.

Table 3.1: DP operators of each layer.

Layer	DP Operator
APP	$V_2\left(s_1', s_2'\right) = \underset{a_3, Z_3}{max}\left[R_{in}\left(s_3, Z_3\right) - \lambda_3^a c_3\left(s_3, a_3\right) + \gamma \sum_{s_3' \in \mathcal{S}_3} p\left(s_3' \middle| s_3, Z_3, a_3\right) V\left(s_1', s_2', s_3'\right)\right]$
MAC	$V_1\left(s_1'\right) = \underset{a_2}{max}\left[-\lambda_2^a c_2\left(s_2, a_2\right) + \sum_{s_2' \in \mathcal{S}_2} p\left(s_2' \middle| s_2, a_2\right) V_2\left(s_1', s_2'\right)\right]$
PHY	$V\left(s_1, s_2, s_3\right) = \underset{a_1}{max}\left[-\lambda_1^a c_1\left(s_1, a_1\right) + \sum_{s_1' \in \mathcal{S}_1} p\left(s_1' \middle| s_1, a_1\right) V_1\left(s_1'\right)\right]$

3.2.3 System Model

The transmission model considered in this dissertation considers a one-hop transmission between the wireless client and the server. Three layers, the PHY layer, the MAC layer, and the APP layer are used to demonstrate the cross-layer optimization process. Optimization in other layers can be established for various wireless networks analogously.

3.2.3.1 PHY Layer Model

At PHY layer, the SINR experienced by the wireless user can be modeled as a discrete time finite state markov chain (FSMC) as described in [22, 47]. Here SINR is used as the PHY layer state s_1^k. The received SINR range is divided into $r+1$ regions with thresholds of $\Gamma_0, \ldots, \Gamma_{r+1}$, where $\Gamma_0 < \Gamma_1 < \cdots < \Gamma_{r+1}$. The PHY layer is said to be in state $s_1^k = \tilde{\Gamma}_i$ if the SINR of the received packet lies in the range of $[\Gamma_{i-1}, \Gamma_i)$, where $\tilde{\Gamma}_i$ is the representative channel gain of this range. State transition is driven by the allocated transmission power set by PHY layer external action $a_1 \in A_1$. The cost incurred by taking an external action a_1^k is $c_1\left(s_1^k, a_1^k\right) = a_1^k$. One-step transition for the PHY layer states is assumed in order to make the model closer to reality. Hence only the transitions between adjacent states are allowed in this layer.

The transmission channel is model as a time-varying Rayleigh fading channel as described in [47]. The channel gain of the Rayleigh fading channel has an exponential distribution with a probability density function expressed in Formula 1.8

$$p(\gamma) = \frac{1}{\gamma_0} exp\left(-\frac{\gamma}{\gamma_0}\right), \gamma \geq 0 \tag{3.9}$$

where γ_0 is the average received SINR, which is determined by the transmission power set by external action a_1. The state transition probability is then calculated in Formula 1.9 as in [47].

$$p(s_1'|s_1, a_1) = \begin{cases} N\left(\tilde{\Gamma}_{i+1}\right)\frac{T_p}{\omega_i}, & s_1 = \tilde{\Gamma}_i, s_1' = \tilde{\Gamma}_{i+1} \\ \mathcal{N}\left(\tilde{\Gamma}_i\right)\frac{T_p}{\omega_i}, & s_1 = \tilde{\Gamma}_i, s_1' = \tilde{\Gamma}_{i-1} \\ 1 - \mathcal{N}\left(\tilde{\Gamma}_{i+1}\right)\frac{T_p}{\omega_i} - \mathcal{N}\left(\tilde{\Gamma}_i\right)\frac{T_p}{\omega_i}, & s_1 = \tilde{\Gamma}_i, s_1' = \tilde{\Gamma}_i \\ 0, & o.w. \end{cases} \tag{3.10}$$

where $\mathcal{N}(\Gamma) = \sqrt{\frac{2\pi\Gamma}{\gamma_0}} f_d exp\left(-\frac{\Gamma}{\gamma_0}\right)$ and $\omega_i = exp\left(-\frac{\Gamma_i}{\gamma_0}\right) - exp\left(-\frac{\Gamma_{i+1}}{\gamma_0}\right)$, with f_d representing the maximum Doppler frequency caused by the wireless user's motion, and T_p the transmission time for one packet.

By adapting the modulation and channel coding scheme via taking internal action $b_1 \in B_1$, the PHY layer provides the MAC layer with an optimal QoS.

3.2.3.2 MAC Layer Model

MAC layer is also modeled as a FSMC with the state $s_2^k \in [0, 1]$ representing the amount of time that the wireless channel is allocated to the service provider. MAC

layer external action a_2^k is defined as service provider's competition bid for the acquisition of spectrum usage. Depending on the type of channel access method, the wireless service provider may or may not take this external action. For TDMA-based channel access, the MAC layer performs external action whenever the service provider requests for the channel usage. The external cost introduced by this action is $c_2\left(s_2^k, a_2^k\right) = a_2^k$. The state transition probability is $p\left(s_2^{k+1}|s_2^k, a_2^k\right)$. As for CDMA-based channel access, the whole channel is available. Thus the MAC layer does not need to request for accessing the channel. As a result, the MAC layer state s_2^k is always 1, and the external action $a_2^k = \varnothing$. There is no external action cost generated in this case. The state transition probability becomes $p\left(s_2^{k+1} = 1|s_2^k = 1, a_2^k = \varnothing\right) = 1$.

The control mechanism, Automatic Repeat reQuest (ARQ), is used at MAC layer to enhance the QoS provided to upper layer at the receiver by requesting the transmitter to retransmit a block of data when errors are detected. The retransmission limit is determined by MAC layer internal action $b_2^k \in \{0, \ldots, N_{max}\}$, where N_{max} is the maximum retry limit.

3.2.3.3 APP Layer Model

At APP layer, different types of data are generated from time to time for various applications. We divide the incoming data into three categories, delay-sensitive data, throughput-constrained data, and cost-restricted data, based on QoS requirements of each application.

When delay-sensitive data are generated, we assume each packet has a lifetime of J stages. After J stages, the packet will be expired. If it is not successfully transmitted during its lifetime, it will be dropped. $s_3^k = \left[s_{3,1}^k, \ldots, s_{3,J}^k\right]^T$ represents the APP layer state at transmission stage k. The external action a_3^k determines the number of newly generated packets to be transfered at APP layer in the beginning

of stage k. Assume a_3^k equals the average number of new packets, the actual number of newly generated packets at stage k is a random variable $Y_3^k\left(a_3^k\right)$. The probability mass function of the random variable is assumed to be independent at each stage, and is denoted by $P\left(Y_3^k = y|a_3^k\right)$, where $y \in \mathbb{N}$.

When the QoS Z_3^k is provided to the APP layer, the number of packets that can be transmitted is calculated as follows:

$$n_3^k\left(Z_3^k\right) = \left\lfloor \frac{\triangle T}{\tau_3^k}\left(1-\varepsilon_3^k\right)\right\rfloor \tag{3.11}$$

where τ_3^k is the transmission delay of each packet in APP layer at stage k, ε_3^k is the packet loss rate in APP layer at stage k. Therefore, the next state of APP layer at stage k+1 is computed as

$$\begin{bmatrix} s_{3,1}^{k+1} \\ \vdots \\ s_{3,j}^{k+1} \\ \vdots \\ s_{3,J}^{k+1} \end{bmatrix} = \begin{bmatrix} s_{3,2}^k - max\left(n_3^k - s_{3,1}^k, 0\right) \\ \vdots \\ s_{3,j+1}^k - max\left(n_3^k - \sum_{m=1}^{j} s_{3,m}^k, 0\right) \\ \vdots \\ Y_3^k\left(a_3^k\right) \end{bmatrix} \tag{3.12}$$

The state transition probability is calculated as in Equation 3.13.

$$p\left(s_3^{k+1}|s_3^k, a_3^k, Z_3^k\right) = \begin{cases} P\left(Y_3^k\left(a_3^k\right) = y\right), & if \text{Equation 3.12} holds \\ 0, & o.w. \end{cases} \tag{3.13}$$

For other applications that have transmission accuracy and transmission power as constraints, the information waiting to be transmitted may not have a strict deadline. In this case, the APP layer is modeled differently. An incoming data buffer with a capacity of buffersize is maintained at the wireless transmitting device. Excessive packets are dropped if the buffer is full. In this case, the next state of the APP layer becomes $s_3^{k+1} = min\left(s_3^k - n_3^k + Y_3^k\left(a_3^k\right), buffersize\right)$.

3.2.4 The Alternative Autonomous Cross-Layer Optimization

The alternative autonomous CLD [16] adopts the basic concepts and the FSMC models of each layer from the original autonomous CLD. To ensure that the cross-layer optimization satisfies different QoS requirements of multimedia communications, and makes quick response to wireless user's application switch, two major modifications were made in the alternative aproach.

- In the original design, when calculating the system reward, although both the packet loss probability and the packet transmission delay were used, they were only used for calculating the throughput. This throughput was the only part that was considered positive contribution to the internal reward. Whereas in the alternative approach, the value generated by transmission delay of delivered packets is explicitly added to the internal reward to complete the calculation of system reward.
- Different applications have different QoS requirements. When forming the QoS frontier set to the upper layer, instead of using the same algorithm as in the original design, the alternative design reduces the amount of QoS levels to be promoted to the upper layer by giving different weights to the QoS elements, and only provide the QoS levels that have better overall credits.

3.2.4.1 New QoS Frontier Computation

Wireless multimedia communication consists of applications with different QoS requirements. Some are delay sensitive, some are throughput constrained, and others are cost restricted. According to the type of application, different weights – $w_\varepsilon, w_\tau, w_\upsilon$– are given to the three parts of QoS –packet loss rate, packet transmission latency, transmission cost– as inputs to the optimization algorithm. Larger weight emphasizes higher requirement on the corresponding QoS part. When computing the QoS frontier set of layer l, if one of the three weights is larger than the other two, the QoS levels that have smaller value of corresponding part get to be included in the QoS frontier set. Otherwise, all three parts of the QoS are taken into consideration. The criteria of whether a QoS level belongs to the QoS frontier set is given in Criteria 1.

Criteria 1. Assume $\mathcal{Z}_l$ is already included in the QoS frontier set of layer l. To determine if a new QoS $\mathcal{Z}'_l$ should be included in the frontier set, the following steps are used.

1. Compare the weights given to each part of QoS, if one is larger than the other two, then $\mathcal{Z}'_l$ is included in the frontier set if its corresponding part of QoS with larger weight is smaller than that of $\mathcal{Z}_l$.

2. If there is not a largest weight out of the three, the overall credit of weighted $\mathcal{Z}'_l$ ($\mathcal{Z}'_{l,w}$) and the weighted $\mathcal{Z}_l$ ($\mathcal{Z}_{l,w}$) are computed, and the values are compared as in Equation 3.14. The result of the comparison (denoted by $DiffQoS$) determines whether $\mathcal{Z}'_l$ belongs to the QoS frontier set too. If $DiffQoS < 0$, $\mathcal{Z}'_l$ will be included in the QoS frontier set, otherwise, it will not be included.

$$\begin{aligned} DiffQoS &= \frac{\mathcal{Z}'_{l,w}-\mathcal{Z}_{l,w}}{\mathcal{Z}_{l,w}} \\ &= \frac{\mathcal{Z}'_l.\varepsilon_l-\mathcal{Z}_l.\varepsilon_l}{\mathcal{Z}_l.\varepsilon_l} \times w_\varepsilon + \frac{\mathcal{Z}'_l.\tau_l-\mathcal{Z}_l.\tau_l}{\mathcal{Z}_l.\tau_l} \times w_\tau \\ &\quad + \frac{\mathcal{Z}'_l.\upsilon_l-\mathcal{Z}_l.\upsilon_l}{\mathcal{Z}_l.\upsilon_l} \times w_\upsilon \end{aligned} \tag{3.14}$$

With these criteria, the alternative CLD never gets more QoS levels in the QoS frontier set than the original CLD does, regardless of the values of the three weights. With the original CLD, as long as there is one part in the QoS smaller than all the current members of QoS frontier set, this QoS gets included in the frontier set too. Whereas in the alternative CLD, even in the second condition in Criteria 1, at least one of the QoS parts needs to be smaller than all the current members in order for this QoS to be included in the frontier set.

3.3 Algorithm Simulation and Results

Simulations of both the original autonomous CLD and the alternative one were performed, and the results were presented to compare the performance of the two CLDs.

3.3.1 Simulation Parameters

The known parameters used in the simulations are listed in Table 3.2.

At PHY layer, in the calculation of ε_1, BER stands for Bit Error Rate, which is calculated as follows:

$$BER = \frac{1}{m} \times erfc(\sqrt{s_1} sin\frac{\pi}{2^b}) \tag{3.15}$$

Table 3.2: Simulation parameters at each layer.

Layer	Parameters	Parameter Values
APP	Maximum queue size	$MaxQueueSize = 8192B$
	APP state	$s_3 \in [0, 8192]$
	Data transfer rate (external action)	$a_3 \in \{1, \ldots, 5\}$
	Trade-off parameter	$\lambda_g = 0.1$
MAC	MAC state	$s_2 = 1$
	Maximum retransmission limit (internal action)	$N_{max} = 5$
	Competition bids (external action)	$a_2 \in \{0, 1\}$
	Trade-off parameter	$\lambda_2 = 1$
PHY	Channel model parameters	$f_d = 50Hz$, $T_p = 0.8ms$, $s_1 \in [109, \ldots, 145]\, dB$
	Modulation level (internal action)	$m \in \{1, \ldots, 4\}\, (BPSK, QPSK, 8PSK, 16PSK)$
	Power allocation (external action)	$a_1 \in \{16, \ldots, 34\}\, dBm$
	Packet loss probability	$\varepsilon_1 = 1 - (1 - BER)^{\eta}$, $\eta = 4$
	Transmission time per packet	$\tau_1 = \frac{T_p}{m}$

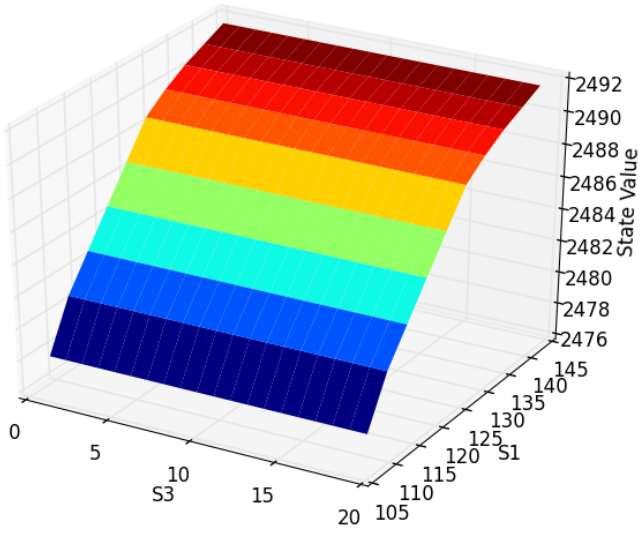
where $b = 2^m$.

3.3.2 Simulation Results

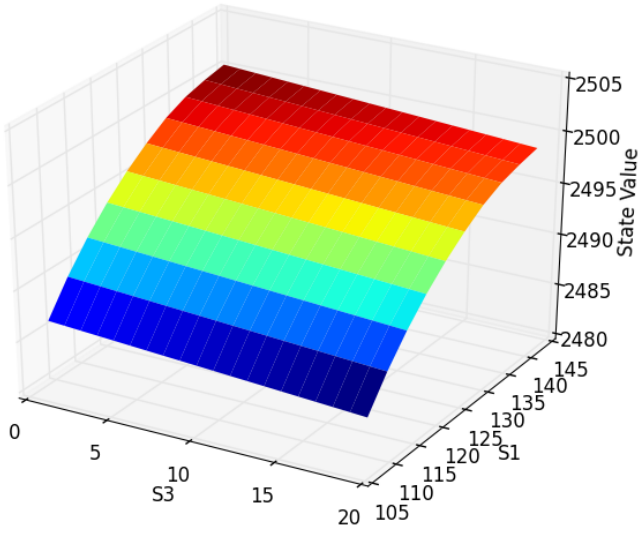
Two experiments were performed to demonstrate the performance of both autonomous CLD methods.

The first experiment performed was simulations of both algorithms with an application that has equal requirements on all three parts of QoS. The achieved state values of all the states are recorded and compared between the two algorithms. The plots in Figure 3.4 show that the alternative autonomous CLD achieves slightly better optimization performance than the original one as the state values obtained are

slightly larger with the alternative autonomous CLD. Different colors in the plots represent different state value ranges.



(a) State value of original autonomous CLD.



(b) State value of alternative autonomous CLD

Figure 3.4: State value of both autonomous CLDs, MAC layer state $s_2 = 1$.

The computation of state values and optimal actions for each layer is performed

prior to applying CLD to wireless transmissions. A lookup table is created after the computation is done. The second experiment was to compare the computation times of creating the lookup tables of the two CLD methods.

The average processing time of calculating each state value was recorded for 10 individual runs for both methods with different types of applications. The results are shown in Table 3.3.

For an application with equally weighted QoS requirements, both methods seem to consume the same amount of time to calculate. However, when the application uses specific weights for QoS requirements, either throughput-sensitive, or cost-restricted, the computation time of the alternative method is reduced almost by 50%. But the original method does not show any difference on computation time. However, with this particular configuration of simulation parameters Table 3.2, the computation time of the alternative CLD for delay-sensitive applications turned out to be the same as if the original CLD was used. It is because the same number of QoS levels were included in the QoS frontier set at the APP layer for both algorithms. In the alternative CLD algorithm, after the possible values of QoS are calculated at the APP layer for each state, the second QoS has the smallest packet transmission delay. According to the criteria of including QoS into the QoS frontier set, the first calculated QoS is always included in the set. Thus two QoS levels are incuded in the frontier set. With the original CLD, according to its criteria of forming the QoS frontier set, it should include at least two QoS levels in the set in this situation. It happened that the following QoS levels did not have any part smaller than the previous two. Hence no more QoS levels were added in the frontier set with the original CLD.

Table 3.3: Computation time comparison of both CLD methods.

Application	Original CLD	Alternative CLD
Equally weighted QoS requirements	4.573 ms	4.36ms
Higher weight for throughput/cost	4.573ms	2.2ms
Higher weight for delay	4.573ms	4.326ms

3.4 Conclusion

Cross-layer optimization has been studied for years to achieve the goal of maximizing network performance by enhancing information exchange between various OSI layers. Previous CLDs usually creates new communication between layers without considering the amount of effort needed to complete the optimization. The autonomous CLD uses a DP operator for each layer in order to make optimization locally. The limited information exchange only happens between adjacent layers, which makes the optimization less resource-extensive.

Wireless multimedia communications require different QoS performance for different applications. Wireless users' requests may vary among different types of information from time to time. It is essential for the source sender to adapt its transmission strategies rapidly according to various requests. The alternative autonomous CLD made some adjustments to the original one to make it fit to the multimedia communications. Simulation results suggest that the alternative autonomous CLD not only preserves the merits that the original autonomous CLD has over regular CLDs, but also outperforms the original autonomous CLD with the capability of adapting to multimedia communications.

Chapter 4

ns-3 Simulation of Alternative Autonomous Cross-Layer Design

4.1 Simulation Purpose

In order to demonstrate the merits of the improved autonomous cross-layer design (CLD), a series of carefully selected simulations have been done with the same network setup for three types of wireless transmissions: regular transmissions without any CLD, transmissions with original autonomous CLD, and transmissions with the alternative autonomous CLD.

To show that the wireless transmissions with alternative autonomous CLD has the best system performance, including system throughput and transmission latency, over the other two types of transmissions, several simulations with a network dealing with fixed size work load were performed. The simulation results shown later in this chapter demonstrate that the alternative design has superior performance.

Next, more simulations are performed to reveal and compare the convergence of

the original and the improved autonomous CLDs. This time a continuous work load for the simulated network is used, so the optimization will not stop before its convergence to a steady state due to short work load. Simulation results suggest that the alternative optimization converges rather quickly, and stays in a stable state for the rest of the simulation time. However, under certain conditions, the original optimization shows the same convergence results. As for actual decisions on the actions to take in each optimization cycle, the original CLD makes sub-optimal decisions when the simulated system is in certain states, whereas the alternative algorithm always makes optimal decisions.

4.2 Experimental Design

4.2.1 Simulation Tool

There are many different network simulators available online. In this dissertation, ns-3 is used as the simulation tool because:

- ns-3 is an open source discrete-event network simulator widely used by researchers and students. It is free to everybody. Users make use of ns-3 by building it from its source code. Thus one can build his/her own ns-3 modules and make use of them freely. When the research in question is more difficult or even impractical to perform with real systems, ns-3 provides the convenience of creating and simulating the system/algorithm rather easier.
- ns-3 is continuously developed by many researchers. Many kinds of communication networks along with different mobility models, application models, etc. are available for use out of box. More models are being added to ns-3 new releases. This means that ns-3 provides better flexibility as for the types of

networks and applications to be simulated.

- Both source code and user program are written in C++, which reduces the need for computing resources and makes simulation run more quickly than other languages (including Java, etc.).
- ns-3 provides a highly controlled, reproducible environment. With appropriate explanations, reproducing an experiment is straightforward.

4.2.2 Network Setup

The network used to do simulation with is an IEEE802.11 Wi-Fi network with one Access Point (AP) and five wireless Stations.

The ns-3 Wi-Fi physical layer uses the Orthogonal Frequency Division Multiplexing (OFDM) modulation scheme to encode data on multiple carrier frequencies, so different transmissions on the same channel should not interfere with each other. However, the ns-3 Media Access Control (MAC) layer does not support concurrent transmissions on a single channel yet, so that a single transmission between the AP and a station node is adopted to simplify the simulation. The AP stays at the same location throughout the simulation, whereas the station node is constantly moving according to a Constant Velocity Mobility Model.

The contributed ns-3 classes YansWifiPhy and YansWifiChannel were used to simulate the physical layer. The channel attributes are set to simulate a Rayleigh Fading channel, which is a typical channel attenuation model for urban communication. Non-QoS-enabled Wi-Fi MAC is used to simplify the simulation and avoid adding unneeded complexity.

4.2.3 Simulated Application

The simulated application mimics a web server with request queueing. The AP acts as a web server who transmits information as requested by a station which acts as a web client. The diagram of the simulated queueing system is shown in Figure 4.1.

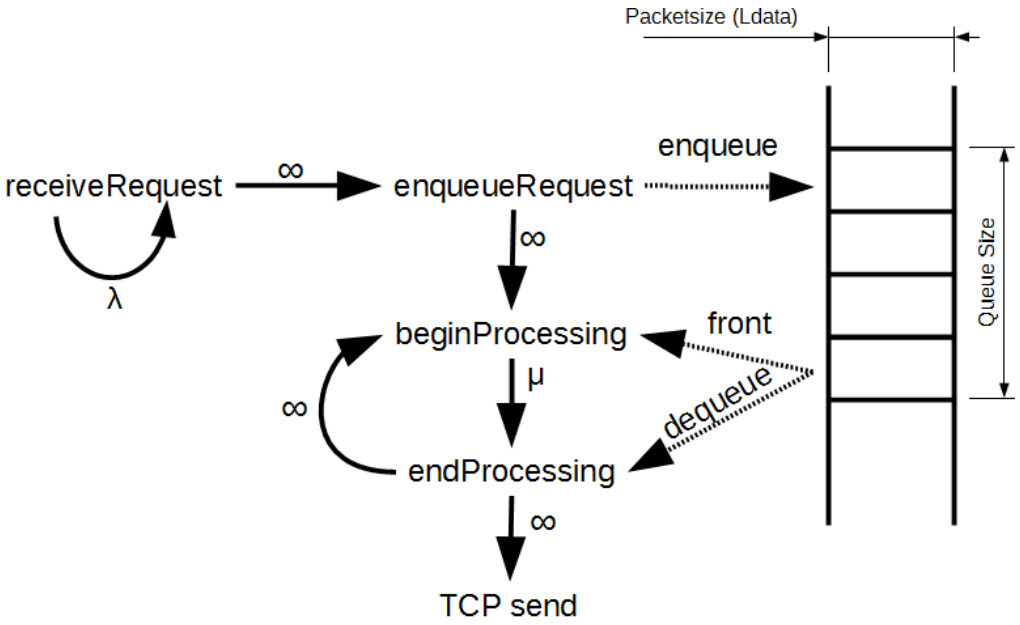


Figure 4.1: Queueing System under Simulation

In the diagram, arrows represent transfer of control to a new functional unit in the simulation and are labeled with the rate of transfer in the state machine. Instantaneous transfers are represented with infinite rate.

A packet flows through this system in the following way. First it is generated by a request generator, which then reschedules itself at the arrival rate λ, to simulate incoming web service requests. The time interval between consecutive incoming packets, namely arrivalInterval (t_{inter}), and the sizes of the packets, namely dataLength (L_{data}), are both modeled as exponential random variables with mean arrival

interval $\overline{t_{inter}}$ and mean data length $\overline{L_{data}}$, where

$$\overline{t_{inter}} = \frac{1}{\lambda} \tag{4.1}$$

The server (AP) maintains a circular queue whose maximum queue size is Q_{max}. The current queue size is the application layer state in the cross-layer optimization. The arrival rate λ and the service rate μ (introduced below) together determine the queue size. When packets are generated, they are entered into the queue and stay there until served. This step is called enqueueRequest in the figure. Random sized packets representing the request response are enqueued in the queue as soon as they arrive. However, when the queue size reaches its maximum, all the incoming packets are dropped without being served. The First Come First Serve (FCFS) queueing discipline is used, so that packets are always stored at the end of queue, and fetched from the front of the queue.

The next event beginProcessing is scheduled immediately in the event enqueueRequest. If it finds that the queue is not busy, the beginProcessing event gets the first packet in the queue and simulates doing processing via passage of time. Afterward it passes the packet to the next event endProcessing. The event endProcessing is scheduled by beginProcessing at a service rate μ, simulating the use of processing time. The total service time t_{serv} includes both the time that the packet spent waiting in the queue for its turn to be served, and the time it spent while being served. It is simulated with an exponential random variable with a mean value of $\overline{t_{serv}}$, where

$$\overline{t_{serv}} = \frac{1}{\mu} \tag{4.2}$$

The serviceTime is modeled as proportional to the size of the packet, as an exponential random variable with a mean value proportional to the dataLength random variable.

In queueing theory, a queueing system is in balance when arrival rate is less than service rate as represented by Equation 4.3.

$$\frac{\lambda}{\mu} < 1 \tag{4.3}$$

Being in balance means that the output rate from the queue is equal to the input rate to the queue. This is shown in our simulations by the simulation plots of queue sizes in the Experimental Results section.

When the last event endProcessing is scheduled, it simulates sending the packet via TCP and dequeues the packet from the queue. It then sets the system to idle state, and schedules another beginProcessing event to fetch more packets if there are any in the queue.

Here are the reasons of using statistical distributions for queueing parameters.

We consider a stochastic model of streaming traffic with requests to the same web server for parts of the same stream. The request arrivals are independent of one another or the state of the system. The probability of an arrival in any interval of time does not depend on the starting point of the arrival or on the specific history of arrivals preceding it. A simple model assumes that the number of arrivals occurring within a given interval of time follows a Poisson distribution. Thus the inter-arrival time is modeled as an exponential distribution.

The authors in [13] analyzed multiple web datasets, and discovered that the file sizes of requests follow the Pareto distribution. Since we were modeling streaming,

however, we used an exponential random variable to represent request sizes. The request size has effect on the service time for the request. The service time is also modeled as an exponential random variable, which does not account for coupling effects between serving several requests.

4.2.4 Experimental Setup

This section gives the detailed explanation about how the simulation is created and setup in ns-3. The types of experiments performed to demonstrate the advanced system performance of the alternative autonomous CLD over the original design and the plain simulation without any CLD are explained next. In the experiments, cross layer optimization is simulated between three OSI layers, application layer, MAC layer, and physical layer.

4.2.4.1 Creating the Simulation Environment in ns-3

Creating ns-3 simulations requires users to write programs in C++ or Python, and link them with the ns-3 software libraries. To make up the network under simulation, ns-3 abstractions–including Node, Application, Channel, Net Device, and Topology Helpers– are used. The way the objects are interconnected is shown in Figure 4.2. [27]

In this implementation of the simulated network, two ns-3 NodeContainers –Apnode and Stanodes– are used to hold the AP node and the Station nodes created next. The ns-3 NetDevice simulates both the hardware network interface card and the software driver. It is strongly bound to the type of Channel that the Node is connecting to through the NetDevice. WifiNetDevices implemented via YansWifiPhy, YansWifiChannel, and WifiMac are considered to be installed on all of the nodes

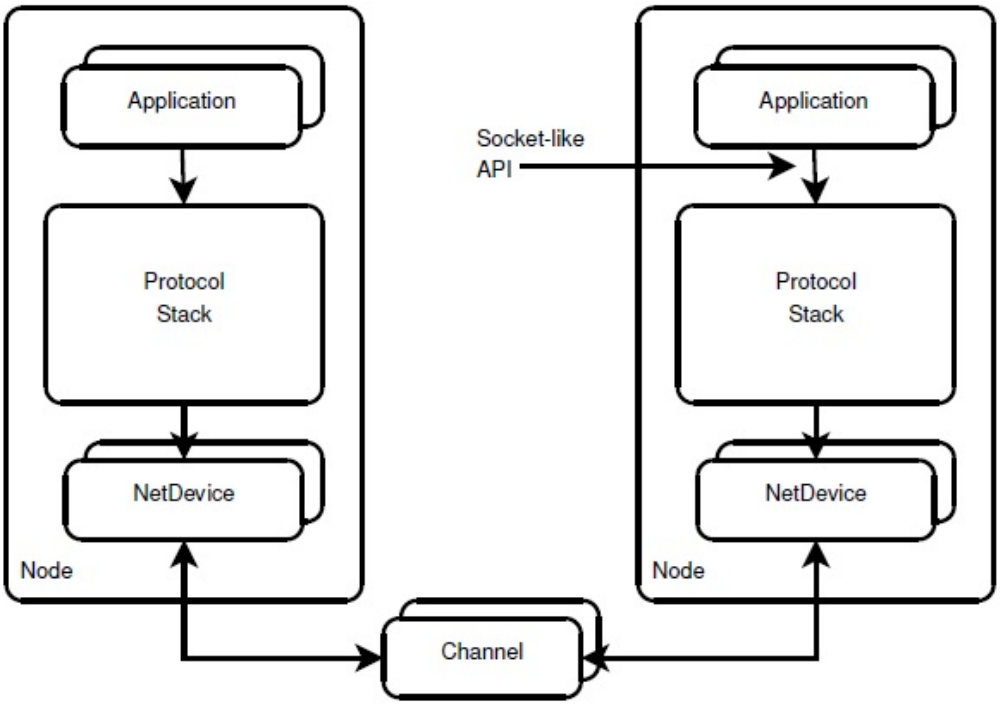


Figure 4.2: Ns-3 Basic Model

in the Wi-Fi network. Applications installed on nodes communicate via protocol stacks. The ns-3 TcpSocket protocol is used for the application under simulation.

There are two approaches to building new applications in ns-3:

- One can work with ns-3 source code and add a new model as ns-3 developers do. This makes permanent change to the ns-3 distribution.
- One can also work in the scratch directory and define a new application along with the main program. This requires less effort than the above method. The ns-3 tutorial provides an example of how to create a new application this way. [5]

The second method is used in the simulation for simplicity. A new application which represents the web application explained above is created and explicitly defined before the main program. After the TcpSocket is created, the application can be installed on the AP node, which can then communicate with the Station node via

the TcpSocket.

Cross layer optimization is defined and initiated before the simulation runs. The optimal actions, both external actions and internal actions, on each OSI layer are recorded for each system state. During simulation, cross layer optimization is scheduled in regular cycles. Each time when the optimization is scheduled, a callback function that executes the change of transmission parameters is invoked.

When executing cross layer optimization, the algorithm tracks current system state, which includes application layer state, MAC layer state, and physical layer state, is needed to obtain the optimal actions for each layer. In the application layer, the queue size is the current state, which is easily obtained from a public function of the queue. Since there is only one transmission on the channel, the MAC layer state in the cross-layer algorithm is set to 1. To obtain physical layer state, the built-in ns-3 tracing system is used. The trace source MonitorSnifferRx provides the custom-implemented trace sink PhyRxTrace with the received signal strength and noise every time a packet is received by the physical layer. This trace sink calculates the average signal to noise ratio over the optimization period, and passes this value as the physical layer state to the optimization function. When all three states are known, the corresponding optimal actions on each layer are properly fetched from the state transition table for the cross layer optimization calculation which was computed prior to simulation.

4.2.4.2 Creating Experiments

Experiments were designed to test whether the alternative autonomous CLD optimizes wireless transmissions greatly and outperforms the original autonomous CLD. Three transmission circumstances are simulated for comparison: transmission without any optimization, transmission with original autonomous CLD, and transmission

with improved autonomous CLD. Two major types of simulations are designed and executed. One with continuous job requests, the other with fixed job sizes.

- From a networking viewpoint, simulations with continuous requests are done to measure the rate of convergence of the optimization algorithms. While keeping the service rate unchanged, different arrival rates and initial states are used in multiple simulations to check the time necessary for convergence to a stable state. When the algorithm chooses the same settings over several cycles, it is deemed to converge to a steady state.
- Transmissions of fixed total size of requests are simulated extensively with different random variable streams over the three transmission circumstances. Average throughput in each circumstance is calculated over multiple runs, and the results are compared to show the throughput advantage of using the improved autonomous CLD on fixed total size of transmission. A delay sensitive Wi-Fi network is simulated by setting the delay weight the biggest when calculating optimal actions with the improved autonomous CLD. After multiple runs, the average packet transmission latency is calculated and compared with the other two transmissions to demonstrate the delay advantage of using the improved autonomous CLD optimization.

4.3 Experimental Results

Before displaying and analysing the experimental results collected during the simulations performed as described in the section above, it is necessary to briefly review the states and the controllable parameters of the algorithm to help understanding the experimental results.

4.3.1 Review of the Autonomous Cross-Layer Optimization Algorithm

Recall from the previous chapter, with the autonomous cross-layer optimization, each OSI layer maintains a dynamic programming (DP) operator and makes transmission optimization locally in the layer. Depending on the current state, each layer selects the locally optimal external action and internal action that maximizes the system reward. These actions set states for the layers. In general, the external action determines the state of each layer, whereas the internal action determines the QoS that the current layer provides to its upper layer.

In this dissertation, three layers were simulated, the PHY layer, the MAC layer, and the APP layer.

In PHY layer, the state was defined as the received signal-to-interference and noise ratio (SINR). The value of external action 1 determines the transmission power of the transmitter. The value of internal action 1 sets the modulation scheme of PHY layer.

MAC layer's state was defined as the percent of time that the transmission occupies the channel. The value of external action 2 is the amount of bids that the sender places in order to get access to the channel. The value of internal action 2 in this layer sets the maximum retransmission limit, which determines the QoS that the MAC layer provides to its upper layer.

The queue size was used as the state of the APP layer. The value of external action 3 in this layer sets the type of source coding scheme the application uses to determine the data transfer rate. There is not an internal action 3 for the APP layer, because it is the top layer in the OSI model, and it does not provide any service to an upper layer.

To ensure clear explanation of the experimental results, a lookup table Table 4.1 is provided to link the terminologies mentioned above to the real parameters in the optimization algorithm.

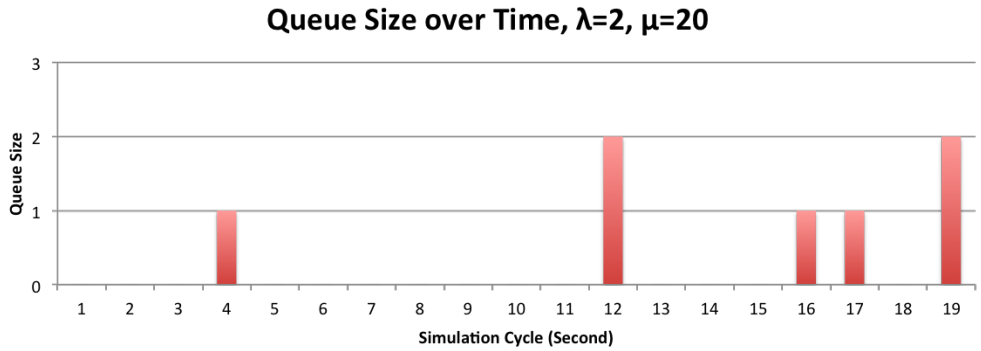
Table 4.1: Lookup table for terminologies.

Layer	Input State	Value of External Action	Value of Internal Action
APP	Queue size.	Source coding mechanism.	N/A
MAC	Percent of time for channel access.	Bid amount for channel access.	Maximum retransmission limit.
PHY	Received SINR.	Transmission power allocation.	Modulation scheme.

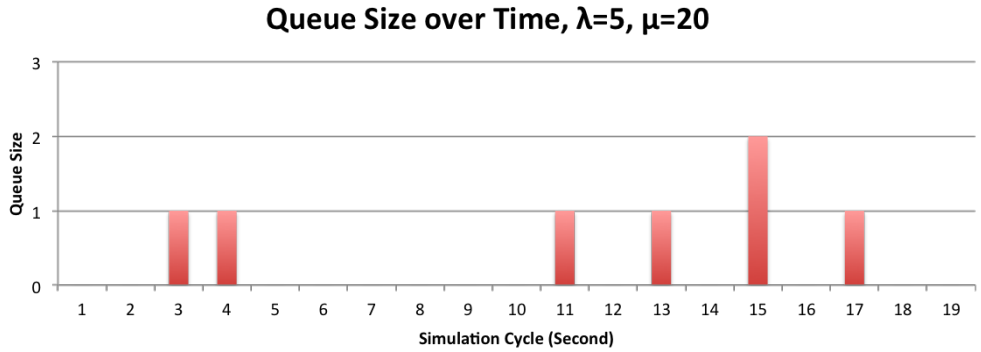
The following sections make detailed analysis over the experimental results.

4.3.2 Continuous Job Convergence Analysis

From a networking viewpoint, it is important to show if the proposed algoritm converges to a stable state after running for a while on a steady workflow. A queueing system with continuous incoming requests fits perfectly for this purpose. Cross-layer optimization was applied once every 1 second. The total simulation time was set to 20 seconds to allow at least 19 optimization cycles to occur.



(a)

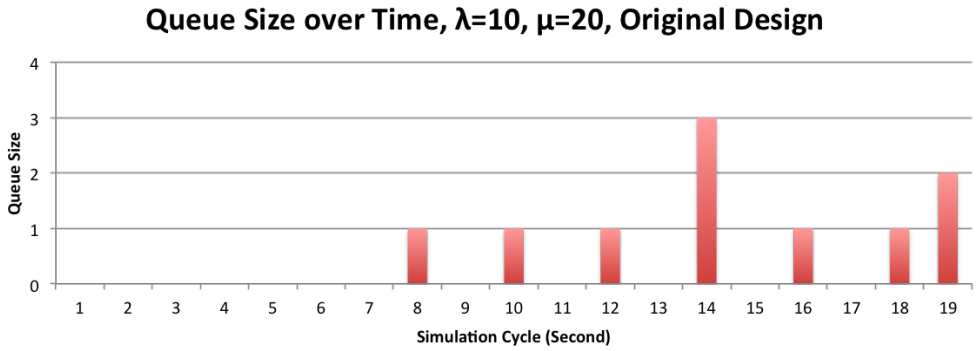


(b)

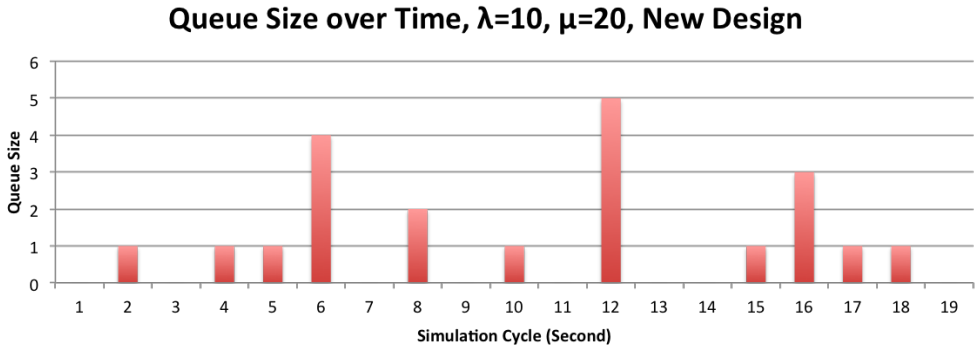
Figure 4.3: Queue size over time for both optimization, λ=2 and 5.

In the first experiment with varying initial arrival rate, the mean service rate μ is set to 20 with mean packet size of 50 bytes. The initial arrival rate λ varies from 0.5 all the way to 20. The application queue size in each simulation cycle is first plotted for different initial arrival rates and optimization configuration, so that the implementation of the application is verified. Then the number of cycles each simulation takes for the system to converge to a stable state is examined.

The plots in Figure 4.3 and Figure 4.4 show that the queue size is always between 0 and 5 when arrival rate λ is smaller than service rate μ. Keeping μ the same, as λ grows bigger, the peak queue size grows. But it always goes back to 0 from non-zero values, which means the application queue is in balance and correctly



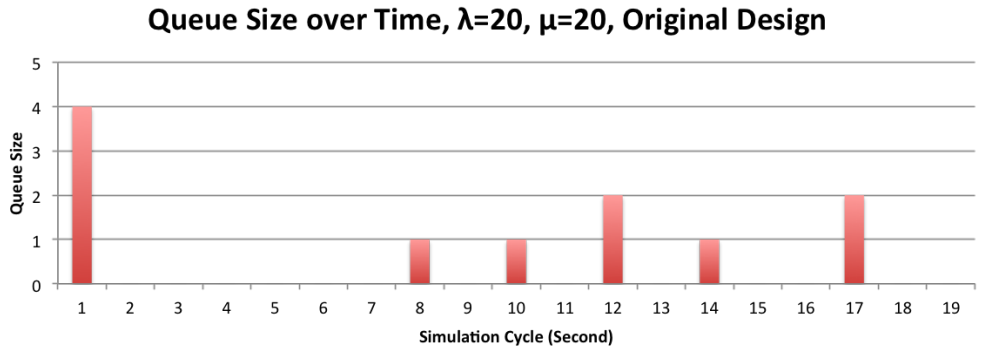
(a)



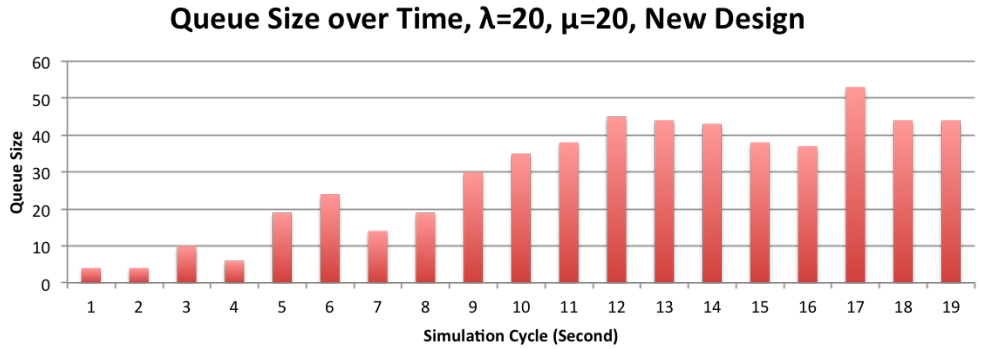
(b)

Figure 4.4: Queue size over time for both optimization, $\lambda = 10$.

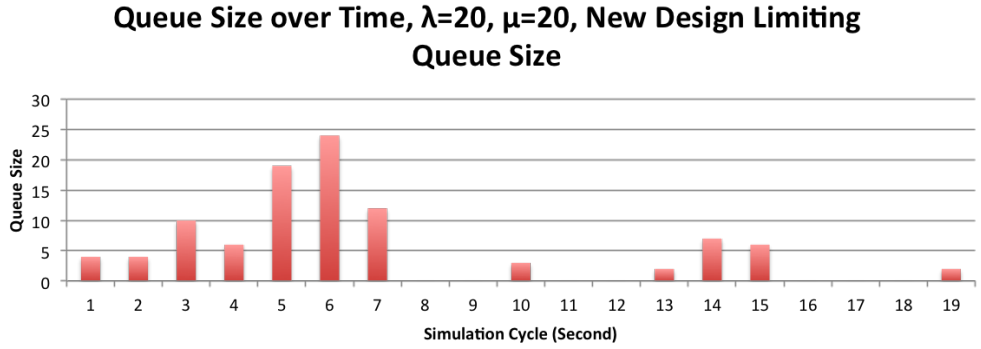
implemented. When $\lambda = 10$, the peak queue size of the new CLD optimization is larger than the original one. This is because in the original CLD optimization, the locally optimal source coding scheme, which determines the new data transfer rate, is limited by the selection range provided by the optimization algorithm. When the initial arrival rate is bigger than the maximum transfer rate allowed by the algorithm, the original optimization changes the transfer rate to its maximum. Whereas in the new optimization, it keeps the transfer rate unchanged if the arrival rate is smaller than the service rate.



(a)



(b)



(c)

Figure 4.5: Queue size over time for both optimization, $\lambda = 20$.

However, with the new optimization, when $\lambda = \mu$, the queue size can grow much larger without returning to 0. Without control, the queue size can keep growing as shown in Figure 4.5(a). When the maximum queue size is reached, all the incoming

packets will be rejected. To prevent this situation from happening, in the new optimization, an extra step that reconsiders a proper source coding scheme is added when the queue is half full. By modifying the optimal source coding scheme in the APP layer, a slower data transfer rate is used. The result in Figure 4.5(b) shows that with this extra control, the queue size stays small and returns to 0.

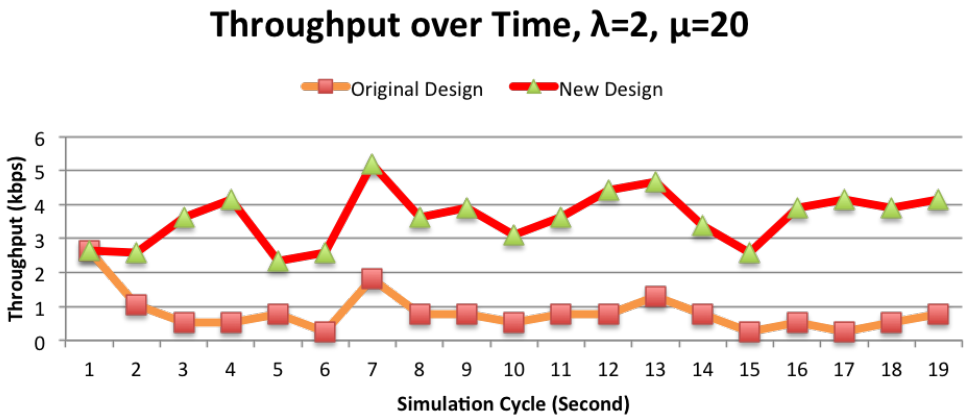
When examining the optimization convergence rate for the algorithms, the following pattern was observed. When optimization starts from the 1st second of simulation, it takes two cycles to converge to a stable state. When it starts from the 2nd second, it only takes one cycle to converge to a stable state. This is because in the 1st second, most communication packets between the AP and the station node are control packets, for example, association packets. These packets are transmitted at higher power, which results in larger received signal strength. Thus the selected transmission power in the 1st optimization cycle is always larger to make sure that the received signal strength does not jump too much between consecutive transmissions.

In the original algorithm, the DP parameter that balances application quality and transmission cost needs to be set smaller in order to get the same settling rate as in the new algorithm. In these simulations, this parameter is reduced for the original algorithm. Since the states do not change too much after the first optimization cycle, according to the algorithms, the locally optimal actions are the same afterwards. The received signal to noise ratio does not vary much, so transmission power stays the same for the rest of the simulation time. MAC layer channel occupancy is always 1 because there is only one transmission happening on the channel. Thus the client does not have to bid for the channel, which means the bid amount for channel access is always 0. Since queue is in balance, both algorithms select the maximum data transfer rate as the closest one to the service rate. The PHY layer modulation scheme affects all three parts of QoS. Both algorithms make the same decision on

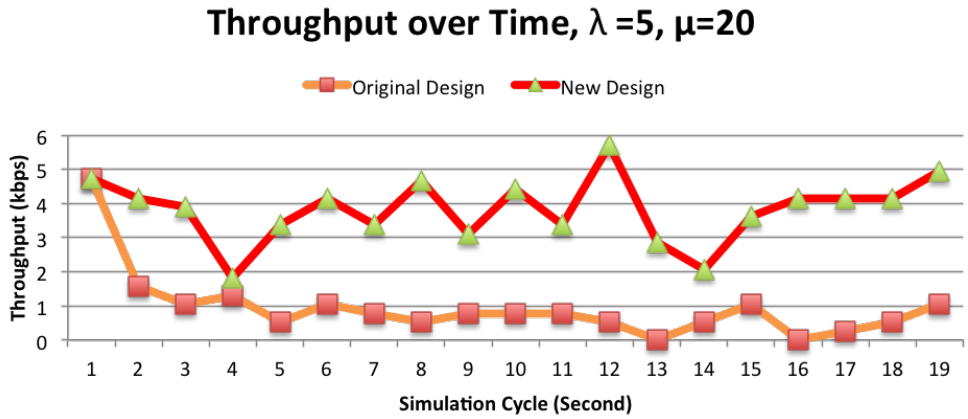
which modulation scheme to use for each optimization cycle.

The second experiment still keeps the mean service time as 0.05 seconds and mean packet size 50 bytes. The mean arrival interval is set to 0.2 seconds. But the initial states of the system vary. The system convergence rate has the same pattern as in the first experiment. Using different initial transmit power, the initial received SINR varies. However, after one optimization cycle, transmit power is changed to a proper value that maximizes system quality and reduces transmission cost. The received SINR does not vary much in following optimization cycles. So the locally optimal transmission power keeps the same according to the same reason that was explained in the paragraph above. When initial queue size is changed, it does not affect the way the algorithms select the locally optimal data transfer rate, because the queue is in balance, the data transfer rate closest to the service rate is always selected to maximize throughput.

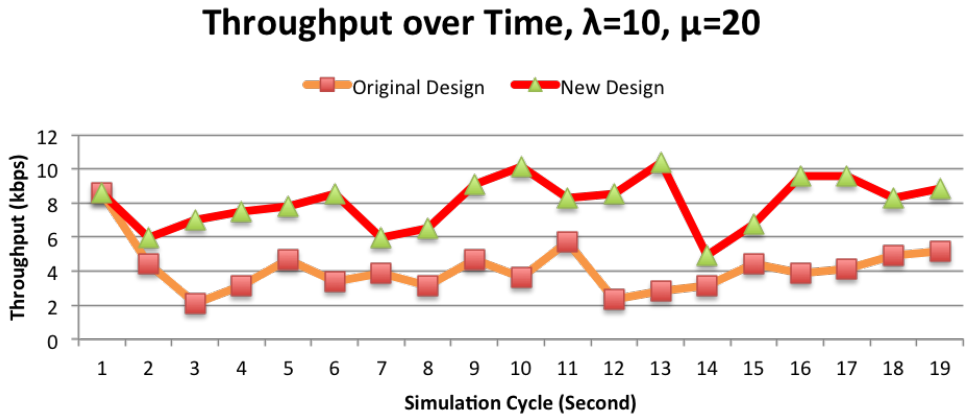
Three different runs of random number generator were done, and the throughput of each optimization cycle was averaged among three runs. The plots in Figure 4.6 demonstrate that with the new design, throughput of each simulation cycle was improved at least by 40% over the original design after the first cycle.



(a) Throughput over time for $\lambda = 2, \mu = 20$.



(b) Throughput over time for $\lambda = 5, \mu = 20$.

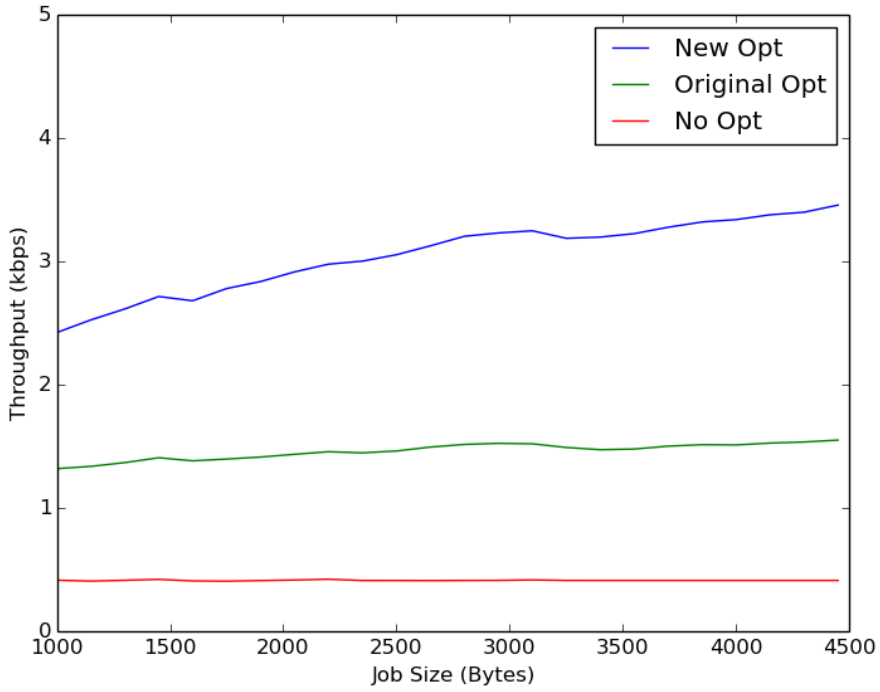


(c) Throughput over time for $\lambda = 10, \mu = 20$.

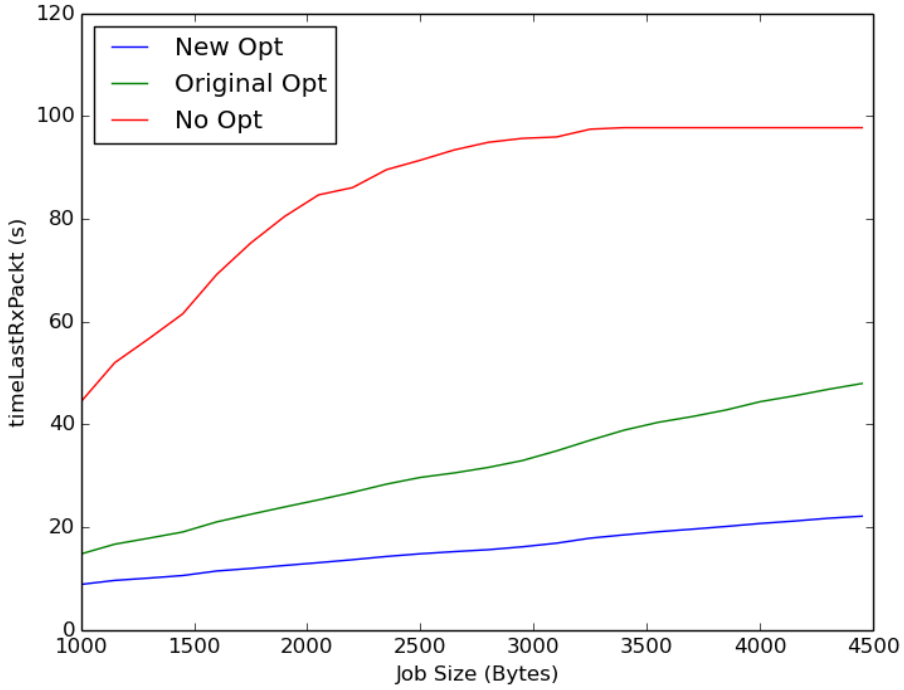
Figure 4.6: Throughput in each simulation cycle for both original design and new design.

4.3.3 Fixed Job Size Throughput Analysis

Five different runs of random variable streams are used in the simulations with different job sizes ranging from 1kB to 4.5kB. When application starts, the mean arrival interval of jobs is set to 2 seconds. The mean service time of the system is always set to 0.05 second. The mean dataLength of each packet is set to 50 bytes. To allow as much data to be transfered during the simulation, a total simulation run time of 100 seconds is used. Optimization was still applied once every 1 second in the simulations. The average throughputs during the transmissions and the times of the last packet received by the station are calculated and recorded. The comparisons among the three transmission circumstances are plotted in Figure 4.7.



(a) Average throughput vs. job sizes.



(b) Job completion time vs. file sizes.

Figure 4.7: Throughput and total delay performance comparison with fixed job sizes.

These figures clearly demonstrate that with the new autonomous CLD, wireless transmission provides better throughput solution, which in turn completes transmission much faster. Because with the new design, the APP layer always selects the maximum external action, which in turn results in fastest data transfer rate. The throughput plot for the simulation without optimization goes flat when the job size is bigger than 3kB. This is because without optimization, the APP layer transfer rate keeps the same, which in this case if much slower than the service rate. As a result, the APP layer is not able to finish transfering the complete job in limited simulation time.

4.3.4 Fixed Job Size Latency Analysis

One of the advantages that the new autonomous CLD over the original design is it fits networks of various QoS needs nicely. Provided with a weight for each part of QoS, the new design is capable of improving the system under optimization toward the required QoS need. To demonstrate this advantage, multiple simulations are done with a delay weight of 0.8, the throughput weight and the cost weight being 0.1. Again five different runs of random variable streams are used in the simulations with different job sizes ranging from 1kB to 4.5kB. Plot of average packet transmission latency is generated from the simulation results. Figure 4.8 demonstrates that with the improvement mentioned above, the new CLD outperforms the original design.

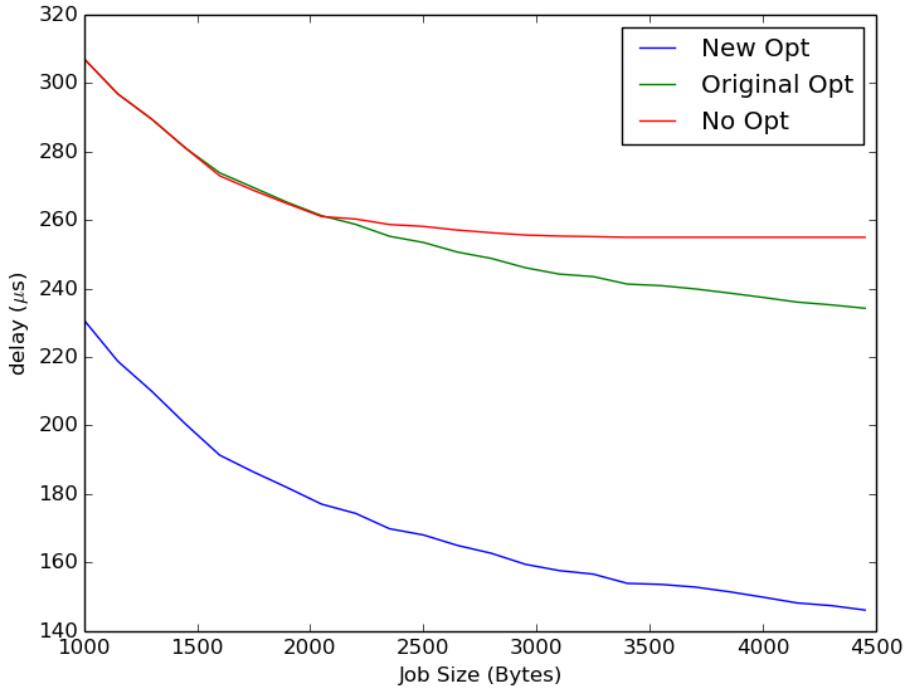


Figure 4.8: Transmission latency comparison with different job sizes.

4.4 Conclusion

This chapter described in detail about the simulations performed to demonstrate the performance improvement that CLD achieves. The comparison between the new autonomous CLD and the original autonomous CLD was also performed with multiple simulations. The results suggest that the new autonomous CLD achieves better performance improvement than the original one, on both throughput and transmission delay. The new autonomous CLD also provides the flexibility of adjusting to different application QoS requirements. By giving different weights for the QoS elements in the algorithm, the optimizer provides optimal actions accordingly. The simulations performed with fixed job size proved this advantage over the original CLD.

Chapter 5

Conclusion

5.1 Thesis Contribution

This thesis discussed about the challenges of designing reliable and efficient wireless networks with emphasis upon cross-layer optimization.

In Chapter 2, one existing cross-layer optimization method was applied to an energy efficient routing algorithm for wireless sensor networks. By retrieving a sensor node's energy level at the Physical layer, the Network layer makes decisions on which neighbor nodes to choose as next hop nodes. At the Physical layer, the destination of data transmission is then set to the designated next hop node's address. Simulation results demonstrated that with this cross-layer optimization, the routing algorithm was found to be more energy efficient; the network lifetime was greatly extended with more distributed energy consumption over the whole network. Simulations also showed that using multiple sinks can further reduce power requirements.

An autonomous cross-layer optimization was studied and introduced in Chapter 3. With this cross-layer optimization, each layer maintains an optimizer and de-

termines its own transmission strategies locally. This overcame the disadvantage of previous cross-layer optimization methods by limiting information exchange to between adjacent layers. A couple of modifications were made to enhence the flexibility of the optimization over multimedia communications, to make the decisions about transmission strategies better optimized, and to make the computation less time consuming. Ns-3 simulations of a web server with request queueing showed that with the modifications, the autonomous cross-layer optimization achieved better network performance by increasing the average throughput and reducing average packet transmission latency.

5.2 Future Work

5.2.1 Energy Efficient Routing Algorithm for WSN

Although the new energy conserving routing algorithm outperforms the original energy efficient routing algorithm by extending the network lifetime by 40%, it can be improved in other ways.

Using multiple sink nodes running at the same time to further distribute the power consumption was discussed in Chapter 2. However, the lower-hop-count sensor nodes can still deplete their energy resources more quickly than other sensor nodes. Activating sink nodes placed at different locations at different time could solve this problem, as it changes the hop counts of sensor nodes.

In the simulation, the sink nodes are always placed in the center of the area of interest. This is not always true in reality. For example, in the application of border surveillance, it is not power efficient to have a sink node right at the border line. Different simulations with different locations of sink node need to be done to check

the flexibility of the routing algorithm for different sink locations.

5.2.2 Autonomous CLD

To solve the autonomous cross-layer optimization in each layer, dynamic programming using value iteration was adopted instead of linear programming. An optimization using linear programming is guaranteed to be convex, but an optimization using dynamic programming is not necessarily convex. Therefore, whether the optimal transmission strategies determined by the optimizers are local optimal values or global optimal values is left to be determined.

In seeking a method to demonstrate the performance improvement that the alternative autonomous CLD makes over the original design, some research was done on how to implement CLD on real hardware. Intel Wireless WiFi Link 5300 (IWL5300) [2] wireless network interface card was used to explore the possiblity of implementing CLD on it. The IWL5300 card supports multiple modulation schemes and transmission powers. It could be a perfect fit for this purpose. The Linux 802.11n CSI tool [3] was used to obtain channel state information of communications using the IWL5300 cards.

However, the only stable version of Linux that supports this CSI tool is Ubuntu 10.04 LTS which uses an old Linux kernel. It was difficult to find computers on which to install the required device driver in the old OS. The CSI tool did not always work as expected. Furthermore, to switch the modulation scheme or transmission power, the wireless card needs to be re-enabled after corresponding parameters have been changed. This requires re-association between communication pairs, which incurs significant delay during transmission. Under this circumstance, CLD obviously might not improve network performance with current hardware in the situation where many state changes are needed. Instead, due to cost of changing

state, it might even reduce network performance. For these reasons, ns-3 simulation was used to demonstrate the advantages of CLD on wireless networks, and the implementation on real hardware was left to future work.

Although it is not a better choice to check the benefits of CLD over currently available hardware alone, it is possible to combine ns-3 simulation with real hardware to partially demonstrate network performance improvement obtained via applying CLD. The ns-3 wiki page [1] provides instructions for connecting ns-3 simulations to the real world. When CLD is implemented in ns-3 and connected to a real network, by checking the network performance in the real world before and after CLD is simulated, the benefits of using CLD can be revealed.

Most people use the OSI reference model as guidance for developing networked applications, but it is easy to forget that the reference model is only a guide and not a proscriptive standard. In this thesis, we have explored the advantages of departing from the suggested boundaries between layers in the ISO reference model. We have shown that in two cases, this departure has performance advantages. This thesis is thus a step toward networking models beyond the confines of the ISO model, that in specific cases have tangible and attainable performance advantages.

Bibliography

[1] S. Shakkottai, T. Rappaport, and P. Karlsson, "Cross-layer design for wireless networks," *IEEE Communications Magazine*, vol. 41, no. 10, pp. 74–80, 2003. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1235598

[2] M. van der Schaar and S. S. N, "Cross-layer wireless multimedia transmission: challenges, principles, and new paradigms," vol. 12, no. 4, pp. 50–58, 2005. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1497858

[3] B. Fu, Y. Xiao, H. Deng, and H. Zeng, "A survey of cross-layer designs in wireless networks," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 1, pp. 110–126, First 2014.

[4] V. Srivastava and M. Motani, "Cross-layer design: A survey and the road ahead," *IEEE Communications Magazine*, 2005.

[5] F. Fu and M. van der Schaar, "A new systematic framework for autonomous cross-layer optimization," vol. 58, no. 4, pp. 1887–1903, 2009. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4663878

[6] Y.-J. Chang, F.-T. Chien, and C.-C. Kuo, "Cross-layer QoS analysis of opportunistic OFDM-TDMA and OFDMA networks," vol. 25, no. 4, pp.

657–666, 2007. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4205049

[7] Y. Dong, C. H. Chang, Z. Zou, and S. Tang, "An energy conserving routing algorithm for wireless sensor networks," *International Journal of Future Generation Communication and Networking*, vol. 4, no. 1, pp. 39–53, 2011.

[8] M. A. Matin, "Wireless sensor networks-technology and protocols," *InTech, September*, 2012.

[9] M.-T. Sun and A. R. Reibman, *Compressed Video over Networks*, 1st ed. New York, NY, USA: Marcel Dekker, Inc., 2000.

[10] X. Wang, Q. Liu, and G. Giannakis, "Analyzing and optimizing adaptive modulation coding jointly with ARQ for QoS-guaranteed traffic," vol. 56, no. 2, pp. 710–720, 2007. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4138054

[11] T. Holliday and A. Goldsmith, "Optimal power control and source-channel coding for delay constrained traffic over wireless channels," in *Communications, 2002. ICC 2002. IEEE International Conference on*, vol. 2, 2002, pp. 831–835. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=996972

[12] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *1st ACM International Workshop on Wireless Sensor Networks and Applications*, 2002, pp. 88–97.

[13] D. C. Steere, A. Baptista, D. McNamee, C. Pu, and J. Walpole, "Research challenges in environmental observation and forecasting systems," in *6th Annual International Conference on Mobile Computing and Networking*, 2000, pp. 292–299.

[14] S. Intille, "Designing a home of the future," vol. 1, no. 2, pp. 76–82, 2002. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1012340

[15] C. D. Kidd, R. Orr, G. D. Abowd, C. G. Atkeson, I. A. Essa, B. MacIntyre, E. D. Mynatt, T. Starner, and W. Newstetter, "The aware home: A living laboratory for ubiquitous computing research," in *2nd International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture*, 1999, pp. 191–198.

[16] L. Schwiebert, S. K. Gupta, and J. Weinmann, "Research challenges in wireless networks of biomedical sensors," in *7th Annual International Conference on Mobile Computing and Networking*, 2001, pp. 151–165.

[17] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," vol. 40, no. 8, pp. 102–114, 2002. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1024422

[18] J. Al-Karaki and A. Kamal, "Routing techniques in wireless sensor networks: a survey," vol. 11, no. 6, pp. 6–28, 2004. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1368893

[19] M. A. Perillo and W. B. Heinzelman, *Fundamental Algorithms and Protocols for Wireless and Mobile Networks, ch. Wireless Sensor Network Protocols.* CRC Hall, 2005.

[20] J. Rabaey, M. Ammer, J. da Silva, J.L., D. Patel, and S. Roundy, "Picoradio supports ad hoc ultra-low power wireless networking," *Computer*, vol. 33, no. 7, pp. 42–48, 2000. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=869369

[21] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next century challenges: Mobile

networking for "smart dust"," in *5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 1999, pp. 271–278.

[22] Y. Wu, X.-Y. Li, Y. Liu, and W. Lou, "Energy-efficient wake-up scheduling for data collection and aggregation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 2, pp. 275–287, March 2009.

[23] S. Ergen and P. Varaiya, "Tdma scheduling algorithms for sensor networks," University of California, Berkley, Tech. Rep., 2005.

[24] Y. Chen and Q. Zhao, "On the lifetime of wireless sensor networks," vol. 9, no. 11, pp. 976–978, 2005. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1524584

[25] K. Subbu and X. Li, "Sfrp: A selective flooding-based routing protocol for clustered wireless sensor networks," in *Radio and Wireless Symposium (RWS), 2010 IEEE*, 2010, pp. 380–383. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5434131

[26] M. Azim, "MAP: Energy efficient routing protocol for wireless sensor networks," in *Ubiquitous Information Technologies & Applications, 2009. ICUT '09. Proceedings of the 4th International Conference on*, 2009, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5405750

[27] K. Zeng, W. Lou, K. Ren, and P. Moran, "Energy-efficient geographic routing in environmentally powered wireless sensor networks," in *Military Communications Conference, 2006. MILCOM 2006. IEEE*, 2006, pp. 1–7. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4086875

[28] Y. Jin, L. Wang, J.-Y. Jo, Y. Kim, M. Yang, and Y. Jiang, "Eeccr: An energy-efficient m-coverage and n-connectivity routing algorithm under border effects

in heterogeneous sensor networks," vol. 58, no. 3, pp. 1429–1442, 2009. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4531127

[29] M. D. Penrose, "On k-connectivity for a geometric random graph," vol. 15, no. 2, pp. 145–164, July 1999.

[30] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *Ad Hoc & Sensor Wireless Networks*, vol. 1, pp. 89–124, March 2005.

[31] W. Daoyuan, T. Hui, and W. Shuang, "Energy-efficient routing research for wsn," in *International Conference on Wireless Communications, Networking and Mobile Computing.* Shanghai: IEEE, September 2007, pp. 2413–2415.

[32] Q. Wang, M. Hempstead, and W. Yang, "A realistic power consumption model for wireless sensor network devices," in *Sensor and Ad Hoc Communications and Networks, 2006. SECON '06. 2006 3rd Annual IEEE Communications Society on*, vol. 1, 2006, pp. 286–295. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4068131

[33] B. Bougard, F. Catthoor, D. Daly, A. Chandrakasan, and W. Dehaene, "Energy efficiency of the IEEE 802.15.4 standard in dense wireless microsensor networks: modeling and improvement perspectives," pp. 196–201, 2005, design, Automation and Test in Europe, 2005. Proceedings. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1395555

[34] D. V. Djonin and V. Krishnamurthy, "Mimo transmission control in fading channels constrained markov decision process formulation with monotone randomized policies," *Signal Processing, IEEE Transactions on*, vol. 55, no. 10, pp. 5069–5083, 2007. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4305437

[35] M. Goyal, A. Kumar, and V. Sharma, "Power constrained and delay optimal policies for scheduling transmission over a fading channel," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 1. IEEE, 2003, pp. 311–320. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1208683

[36] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov chains and mixing times*. American Mathematical Soc., 2009.

[37] V. Bhaskar, "Finite-state markov model for lognormal, chi-square (central), chi-square (non-central), and k-distributions," *International Journal of Wireless Information Networks*, vol. 14, no. 4, pp. 237–250, 2007. [Online]. Available: http://dx.doi.org/10.1007/s10776-007-0065-2

[38] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2009, vol. 414.

[39] Markov decision process. [Online]. Available: https://en.wikipedia.org/wiki/Markov_decision_process

[40] Q. Zhang and S. Kassam, "Finite-state Markov model for Rayleigh fading channels," vol. 47, no. 11, pp. 1688–1692, 1999. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=803503

[41] Y. Dong, C. H. Chang, and Z. Yang, "An improved autonomous cross-layer optimization framework for wireless multimedia communication in heterogeneous networks," *International Journal of Networked and Distributed Computing*, vol. 2, no. 4, pp. 231–240, October 2014.

[42] M. E. Crovella, M. S. Taqqu, and A. Bestavros, "Heavy-tailed probability distributions in the world wide web," 1998, pp. 3–25.

[43] M. Lacage. (2009) Experimentation with ns-3.

[44] ns-3 tutorial. [Online]. Available: https://www.nsnam.org/docs/release/3.24/tutorial/ns-3-tutorial.pdf

[45] Intel ultimate n wifi link 5300 and intel wifi link 5100 products overview. [Online]. Available: http://www.intel.com/products/wireless/adapters/5000/

[46] Linux 8802.11 csi tool. [Online]. Available: http://dhalperi.github.io/linux-80211n-csitool/

[47] How to make ns-3 interact with the real world. [Online]. Available: https://www.nsnam.org/wiki/HOWTO_make_ns-3_interact_with_the_real_world